QuickWorks® User Manual

with SpDE Reference (Release 2008.2.1)

Contact Information

QuickLogic Corporation
1277 Orleans Drive
Sunnyvale, CA 94089
Phone: (408) 990-4000 (US)
(905) 940-4149 (Canada)
+(44) 1932-57-9011 (Europe)
+(852) 2567-5441 (Asia)
E-mail: info@quicklogic.com
Sales: America-sales@quicklogic.com
Europe-sales@quicklogic.com
Asia-sales@quicklogic.com
Japan-sales@quicklogic.com
Support:www.quicklogic.com/support
Internet:www.quicklogic.com

Notice of Disclaimer

QuickLogic is providing this design, product or intellectual property "as is." By providing the design, product or intellectual property as one possible implementation of your desired system-level feature, application, or standard, QuickLogic makes no representation that this implementation is free from any claims of infringement and any implied warranties of merchantability or fitness for a particular purpose. You are responsible for obtaining any rights you may require for your system implementation. QuickLogic shall not be liable for any damages arising out of or in connection with the use of the design, product or intellectual property including liability for lost profit, business interruption, or any other damages whatsoever. QuickLogic products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use QuickLogic products in these types of equipment or applications.

QuickLogic does not assume any liability for errors which may appear in this document. However, QuickLogic attempts to notify customers of such errors. QuickLogic retains the right to make changes to either the documentation, specification, or product without notice. Verify with QuickLogic that you have the latest specifications before finalizing a product design.

Copyright and Trademark Information

Copyright © 1991-2008 QuickLogic Corporation.
All Rights Reserved.

The information contained in this document and the accompanying software programs is protected by copyright. All rights are reserved by QuickLogic Corporation. QuickLogic Corporation reserves the right to modify this document without any obligation to notify any person or entity of such revision. Copying, duplicating, selling, or otherwise distributing any part of this product without the prior written consent of an authorized representative of QuickLogic is prohibited.

QuickLogic, the QuickLogic logo, pASIC, ViaLink, and QuickWorks are registered trademarks of QuickLogic Corporation. Eclipse, EclipsePlus, SpDE, ArcticLink, PolarPro, and PolarPro II are trademarks of QuickLogic Corporation. All other trademarks or registered trademarks are the properties of their respective owners.

Contents

Chapter 1: Introduction 1

1.1 Product Information 1
1.2 Package Contents 2
1.3 About This Manual 2
1.4 Technical Support 3

Chapter 2: Requirements and Installation 5

2.1 Hardware Requirements 5
2.2 Software Requirements 5
2.3 QuickWorks Installation 6

Chapter 3: Design Overview 13

3.1 SpDE Definitions 13
3.1.1 Key Design Concepts 13
3.2 Design Flows 14
3.2.1 Creating a Schematic-Based Design Flow 14
3.2.1.1 Entering a Design in the Schematic Editor 15
3.2.1.2 Building an In-Memory Database with the Hierarchy Navigator 15
3.2.1.3 Using SpDE 16
3.2.1.4 Programming a Device 16
3.2.2 Creating a Mixed-Mode Design Flow 16
3.2.2.1 Capturing a Design in SCS 17
3.2.2.2 Building an In-Memory Database with the Hierarchy Navigator 18
3.2.2.3 Using the Waveform Editor, Active HDL Simulator, and the Data Analyzer 18
3.2.2.4 Using Precision RTL Synthesis 18
3.2.2.5 Using SpDE, Active HDL Simulator, and the Data Analyzer 18
3.2.3 Creating a Verilog-Only Design Flow 19
3.2.3.1 Entering a HDL Design 19
3.2.3.2 Performing the Prelayout Simulation 19
3.2.3.3 Performing Synthesize, Place-and-Route 19
3.2.3.4 Performing a Postlayout Simulation 20
3.2.4 Creating a VHDL-Only Design Flow 20
3.2.4.1 Entering a VHDL Design 20
3.2.4.2 Performing a Prelayout Simulation 20
3.2.4.3 Using Precision RTL - SpDE 21
3.2.4.4 Performing a Postlayout Simulation 21
3.2.5 QuickWorks Design Flow GUI 22
3.2.5.1 Disabling/Enabling QuickWorks Design Flow 23
3.2.5.2 Using QuickWorks Design Flow 23
3.2.5.3 QuickWorks Design Flow Descriptions 24

© 2008 QuickLogic Corporation

Chapter 4: Command Line Interface 25

4.1 Command Line Options 26
4.1.1 LOAD and IGNORE_CONSTRAINTS 26
4.1.2 RUNTOTOOL and RUNALLTOOLS 27
4.1.3 SAVE 27
4.1.4 BACKANNOTATE_ALL 28
4.1.5 REPORTFILE 28
4.1.6 GET_TOOL_OPTIONS 28
4.1.7 SET_TOOL_OPTIONS 28
4.1.8 POWERCALC 29
4.1.9 RAMROM 29
4.1.10 TIMINGREPORT 29
4.1.11 PATHANALYZER 30
4.1.12 GENHEX 30
4.1.13 TCL 30
4.2 TCL Interface of QuickWorks - CLI 31
4.2.1 Load 31
4.2.2 RunToTool and RunAllTools 32
4.2.3 Save 32
4.2.4 GenerateReport 33
4.2.5 RunBA 33
4.2.6 SetOption and SetOptionSess 33
4.2.7 PathAnalyzer 34
4.2.8 PowerCalc 34
4.2.9 RamRom 34
4.2.10 TimingReport and SetXRPTOption 35
4.3 Running TCL Commands from SpDE GUI 36
4.3.1 Running TCL Script File 36
4.3.2 Running TCL Command from Command Prompt 36
4.3.3 TCL Commands for GUI 36
4.3.3.1 Load 38
4.3.3.2 RunToTool and RunAllTools 38
4.3.3.3 Save 39
4.3.3.4 GenerateReport 39
4.3.3.5 SetOption and SetOptionSess 39
4.3.3.6 PathAnalyzer 40
4.3.3.7 StaticTimingAnalyzer 40
4.3.3.8 PowerCalc 40
4.3.3.9 RamRom 41
4.3.3.10 TimingReport 41
4.3.3.11 GetToolOptions 41
4.3.3.12 SetToolOptions 42
4.3.3.13 SetMacroMode 42
4.3.3.14 Close 42

© 2008 QuickLogic Corporation

4.3.3.15 exit/quit 42
4.3.3.16 Help 42
4.3.4 Usage of Recording Feature in SpDE GUI 42
4.3.4.1 Recording Toolbar 43
4.3.4.2 Start Recording 43
4.3.4.3 Play Recording 43
4.3.4.4 Stop Recording 43
4.3.5 Limitations 43
4.4 Tables 44
4.4.1 Tools Options List 44
4.4.2 Timing Report Options List 45
4.5 File Formats 46
4.5.1 Tool Options File 46
4.5.2 Power Calculator Input File 46
4.5.3 Timing Report Input File 47
4.5.4 Ram Rom Input File 48
4.5.5 Gen Hex Input File 49
4.6 Sample Scripts 51

Chapter 5: Eclipse Devices 53

5.1 Overview of Clock Networks 53
5.2 Dedicated Clock Networks 55
5.3 Global Clock Networks 56
5.4 High Drive Networks 58
5.5 Supported I/O Standards in Eclipse Devices 59
5.5.1 Input Standards 60
5.5.2 Output Standards 60
5.6 Embedded Computational Units (ECUs) 61
5.6.1 Incorporating ECU Modules into Schematic Design 61
5.6.2 Incorporating ECU Modules into HDL-Based Designs 61
5.6.2.1 ECU Models for Designs Using VHDL 62
5.6.2.2 ECU Models for Designs Using Verilog 65
5.7 Configuration Editor 69
5.8 New Pads for Eclipse Devices 70
5.8.1 INPADS 70
5.8.2 I/O Pads 70
5.8.3 HDPAD 71
5.8.4 BIPADS 71
5.8.5 OUTPADs 72
5.8.6 TRIPADs 72
5.9 Phase Lock Loop (PLL) Macros 73
5.9.1 Schematic Entry Example 74
5.9.2 Prelayout Simulation Waveform 74

© 2008 QuickLogic Corporation

5.9.3 Postlayout Simulation Waveform 74
5.9.4 Using PLL Dedicated Pins 75
5.9.5 Disabling PLLs 75
5.9.5.1 PLL_DIV2HF 75
5.9.5.2 PLL_DIV2LF 76
5.9.5.3 PLL_DIV4 77
5.9.5.4 PLL_HF 78
5.9.5.5 PLL_LF 79
5.9.5.6 PLL_MULT2HF 80
5.9.5.7 PLL_MULT2LF 81
5.9.5.8 PLL_MULT4 82

Chapter 6: PolarPro Devices 83

6.1 PolarPro Overview 83
6.2 Features and Benefits 83
6.3 PolarPro Resources 84
6.3.1 PolarPro I/O Banks 85
6.3.1.1 DDRIOs 86
6.3.2 GPIOs 86
6.3.3 Clock Network 87
6.3.4 CCM 88
6.3.5 RAM/FIFO 90
6.3.5.1 RAM 90
6.3.5.2 FIFO Controller 91
6.3.6 Logic Cells 92

Chapter 7: PolarPro II Devices 93

7.1 PolarPro II Overview 93
7.2 Features and Benefits 93
7.3 PolarPro II Resources 94
7.3.1 PolarPro II I/O Banks 95
7.3.2 GPIOs 96
7.3.3 Clock Network 97
7.3.4 CCM 97
7.3.5 RAM/FIFO 99
7.3.5.1 RAM 99
7.3.5.2 FIFO Controller 100
7.3.6 Logic Cells 102

Chapter 8: Power Calculator 105

8.1 Using Power Calculator 105
8.2 Power Consumption Parameters 107
8.2.1 FPGA Power Calculation 107
8.2.2 ASSP Block Power 108

© 2008 QuickLogic Corporation

8.3 Power Calculation Assumptions 110
8.4 Power Calculator Output 110
8.5 TCL/CLI Support 110

Chapter 9: Power Simulator 111

9.1 Functional Overview 111
9.2 Power Simulator Environment Setup 111
9.3 Using Power Simulator 112
9.3.1 Invoking Power Simulator Tool from SpDE 112
9.3.2 Step 1: Creating Simulation Macro and Simulation Task file 112
9.3.3 Step 2: Setting Up Simulation Environment and Activating Simulator 114
9.3.4 Step 3: Generating Power Simulator Output 116
9.3.5 Step 4: Using Power Extrapolation Tool 117
9.4 Limitations 118

Chapter 10: Retarget Devices 119

10.1 Using the Retarget Devices Function 119

Chapter 11: Design Techniques 121

11.1 Using Automatic Buffering Tools 121
11.1.1 How Automatic Buffering Works 121
11.1.1.1 Determining Critical Paths 121
11.1.1.2 Choosing Nets to Buffer 121
11.1.2 Using the SpDE Auto-Buffering Algorithm 122
11.1.3 Disabling SpDE Auto-Buffering Net-by-Net 122
11.2 Evaluating the Results of Auto-Buffering 123
11.2.1 Finding Auto-Buffered Nets in the Path Analyzer 123
11.3 Inserting Schematic Buffers to Speed up the Design 125
11.3.1 Split Buffering 125
11.3.2 Selective Buffering 125
11.3.3 Paralleling 126
11.3.4 Double-Buffering 127
11.4 Inserting Buffers in VHDL and Verilog Designs 128
11.4.1 Instantiate a Special Buffer in Verilog 128
11.4.2 Instantiate a Special Buffer in VHDL 128
11.5 Pipelining to Increase Clock Frequency 130
11.6 Verilog State Machine Design 130
11.6.1 Encoded State Machines 130
11.6.2 One-Hot State Machines 133
11.6.3 Alternative Coding Technique for One-Hot State Machines 135

© 2008 QuickLogic Corporation

Chapter 12: Logic Optimizer 137
12.1 Partition Logic with the Logic Optimizer 137
12.1.1 Level 0: Packer 137
12.1.2 Level 1: Technology Map 138
12.2 Choosing Area or Speed Optimization 139
12.3 Logic Optimization Modes 139
12.4 Using the Pack Attribute on Instances in a Design 140
12.5 Automatic Buffer Insertion 141

Chapter 13: Placer 143
13.1 Window-Based Placer 143
13.2 Placer Options 147
13.2.1 Placer Seed 147
13.2.2 Placer Mode 147
13.2.3 Placer Type 148
13.2.4 Placer with Low Power 148
13.2.5 Gclk Buffer Insertion 148
13.3 Path Constraint-Driven Placement 149
13.4 Fixed Placement 151
13.4.1 Fixing the Placement of I/O Pads 151
13.4.1.1 Manually Assigning Pin Placement in QuickWorks Schematic-Based Designs 151
13.4.1.2 Manually Assigning Pin Placement in QuickWorks Verilog- or VHDL-Only Designs 152
13.4.1.3 Fixing the Placement of Flip-Flops 152
13.4.1.3.1 Fixing Flip-Flops Manually in QuickWorks Schematic-Based Designs 152
13.4.1.3.2 Fixing Flip-Flops and I/O Pins Automatically 153
13.4.1.4 Locking Down a Previous Placement 153
13.5 Timing-Driven Placement 154
13.6 Low Power Placement 154

Chapter 14: Router 155
14.1 Router Tool Options 155
14.1.1 GMUX Selection 155
14.2 Using the Priority Router 156
14.2.1 QCF File Interface 156
14.2.2 Static Timing Analyzer Interface 156
14.2.3 Running Router Tool 157
14.2.4 Router Tool Limitations 157
14.3 Constraint-Driven Routing 157
14.4 Interconnect Resources 158
14.4.1 Array Clock Networks 158
14.4.2 Global Clock Networks for Eclipse and EclipsePlus Devices 158
14.4.3 Global Clock Networks 159

© 2008 QuickLogic Corporation

14.4.3.1 Segmented Wires 161
14.4.3.2 Dual Wires 161
14.4.3.3 Quad Wires 161
14.4.3.4 Express Wires 162
14.4.3.5 Distributed Network Wires 162
14.5 Routing High-Drive Nets 162
14.5.1 High-Drive Pads 162
14.5.2 Parallel Logic 162
14.6 Manual Routing Editor 163
14.6.1 Opening Manual Routing Editor 163
14.6.2 Routing Editor Toolbar 164
14.6.3 Deleting a Route On a Net 165
14.6.4 Adding a Route On a Net 166
14.6.5 Error Messages 167
14.6.6 Miscellaneous Issues 168
14.6.6.1 Detecting Whether Net Has Antenna 168
14.6.6.1.1 Condition for Antenna 168
14.6.6.2 Logic Jog 169

Chapter 15: Back Annotation and Sequencer 171

15.1 Functional Overview 171
15.2 Delay Modeler 172
15.2.1 Delay Modeler Options Tab 172
15.2.1.1 Delay Modeler Operating Range 172
15.2.1.2 Delay Modeler Corner 173
15.2.1.3 Delay Modeler Out-Pad Load 173
15.2.1.4 Delay Modeler Speed Grade 173
15.2.1.5 Delay Modeler Low Power 173
15.3 Custom Temperature and Voltage 174
15.4 Back Annotation 174
15.4.1 Back Annotation Options 174
15.4.1.1 Back Annotation for Verilog or VHDL Header Files 175
15.4.1.2 Verilog 175
15.4.1.3 VHDL (Vital-Compliant, Synopsys VSS, Model Tech V-System, Active HDL) 176
15.4.1.4 Viewsim (ViewLogic) 176
15.4.1.5 EDIF Simulation Netlists (LMC, QuickSim) 176
15.5 Sequencer 177
15.5.1 8-Bit Multibit Sequencer Support for QuickLogic Pinnacle (0.25 µm) Family of Devices 177
15.5.2 4-Bit Multibit Sequencer Support for QL8150 and QL8050 Devices 177

Chapter 16: Design Flows and Reference 179

16.1 Functional Overview 179
16.2 Design Flows 180
16.2.1 Schematic-Based Design Flow 180
16.2.1.1 Entering a Design 181

© 2008 QuickLogic Corporation

16.2.1.2 Adding Top Level I/O Pads to the Design 182
16.2.1.3 Adding Attributes to Symbols in the Design 182
16.2.1.3.1 Setting PLACE and FIXED Attributes 183
16.2.1.4 Pin Placements for Bussed Pads (IPAD4, OPAD8, etc.) 183
16.2.1.5 Creating a Verilog, VHDL, QDIF or EDIF Netlist from the Hierarchy Navigator 184
16.2.1.5.1 Verilog Netlist Format 184
16.2.1.5.2 VHDL Netlist Format 185
16.2.2 Verilog-Only Design Flow 185
16.2.2.1 Entering a Design 185
16.2.2.2 Using the Turbo Writer Editor 186
16.2.2.3 Fixing Pins 186
16.2.2.4 Compiling, Placing, and Routing the Verilog Design 186
16.2.3 VHDL-Only Design Flow 187
16.2.3.1 VHDL Design Flow With Active HDL Simulator or a Third Party VHDL Simulator 187
16.3 SCS Tools Reference 189
16.3.1 Schematic Editor & Navigator 189
16.3.1.1 Creating Buses in the Schematic Editor 190
16.3.1.2 Grounding a Bus 190
16.3.1.3 Connecting Buses 190
16.3.1.4 Using Iterated Instances in the Schematic Editor 190
16.3.2 Symbol Editor 190
16.3.3 Hierarchy Navigator 191
16.3.3.1 Tree (.TRE) and Schematic (.SCH) Files 191
16.3.3.2 Exporting the Netlist 192
16.3.4 Simulation 193
16.3.5 Updating Schematics 193
16.3.6 Changing SCS Design Options 193
16.3.7 Turbo Writer 193
16.3.8 Verilog/VHDL Entry 193
16.3.8.1 Creating a Verilog or VHDL Symbol 194
16.3.8.2 Creating Verilog (.v) and VHDL (.vhd) Source Files 195
16.3.8.2.1 Launching Turbo Writer 195
16.3.8.2.2 Using Context Sensitive Editing 196
16.3.8.2.3 Understanding Color Coding 196
16.3.8.2.4 Using the Syntax Checker 196
16.4 Active HDL Integration 197
16.4.1 SpDE Menu Option 197
16.4.2 Active HDL Interface Inputs 197

Chapter 17: Hard Macros 201

17.1 Functional Overview 201
17.1.1 Terms and Definitions 201
17.2 Macro Creation Design Flow 202
17.2.1 Setting up SpDE for Creating Macros 202
17.2.2 Importing Design into SpDE 202

© 2008 QuickLogic Corporation

17.2.3 **Macro IO Selection** **202**
17.2.4 Processing Macro Design 203
17.2.4.1 Automatic Selection of Window 203
17.2.5 Save Design as Macro 204
17.2.5.1 Save As Macro File Attributes 204
17.2.6 Setting Constraints for Creating a Macro 206
17.3 Using a Macro in a Design 207
17.3.1 Creating Designs Using Macros 207
17.3.1.1 Schematic Based Flow 207
17.3.1.2 HDL Based Flow 208
17.3.2 Functional Simulation 208
17.3.3 Setting Up SpDE for Using Macros 209
17.3.4 Importing Design (Macro + User Design) in SpDE 209
17.3.5 Running Tools 212
17.3.6 Macro Planner 212
17.3.7 Using Macro Planner 213
17.3.7.1 Macro Planner View 213
17.3.7.2 Invoking the Macro Planner View 214
17.3.7.3 Moving Instances 214
17.3.7.4 Macro Planner Menu 216
17.3.7.5 Macro Planner Toolbar 216
17.3.7.6 Rotate 90 Degrees 217
17.3.7.7 Rotate 180 Degrees 217
17.3.7.8 Rotate 270 Degrees 217
17.3.7.9 Flip Horizontally 217
17.3.7.10 Flip Vertically 218
17.3.7.11 Fix Position 218
17.3.7.12 Unfix Position 218
17.3.7.13 Fix Orientation 218
17.3.7.14 Unfix Orientation 218
17.3.7.15 Highlighting Macro Instance 219
17.3.7.16 Saving Placement 219
17.3.8 Macro Fix Placement Constraints 219
17.3.8.1 Macro Orientation 220
17.3.8.2 Macro Constraint Syntax 220
17.3.8.2.1 Constraints for RAM/ECUs Gates 221
17.3.8.2.2 RAM/ECUs In Macro 221
17.3.9 Processing Design 221
17.4 Warning and Error Messages 222
17.4.1 Fatal Errors 222
17.4.2 Errors 222
17.4.3 Warnings 223

© 2008 QuickLogic Corporation

Chapter 18: Creating RAM, FIFO, and ROM Modules 225
18.1 Functional Overview 225
18.2 Creating RAM, FIFO and ROM Modules for Pinnacle, Pre-Pinnacle and Eclipse Devices 225
18.2.1 Creating RAM Blocks 225
18.2.1.1 RAM Design Files 228
18.2.2 Creating FIFO Modules 228
18.2.2.1 FIFO Design Files 231
18.2.3 Creating ROM Modules 232
18.2.3.1 ROM Data Input File 233
18.2.3.2 ROM Design Files 235
18.2.3.3 Using the RAM/ROM/FIFO Components in HDL Designs 236
18.2.4 Asynchronous FIFO Design Functionality 236
18.2.4.1 QuickLogic Asynchronous FIFO Features 236
18.2.4.2 Asynchronous FIFO Overview 237
18.2.4.3 Functional Description and Timing Waveforms 238
18.2.4.4 Utilization and Performance Benchmarks 241
18.2.4.5 Asynchronous FIFO Core Package 242
18.2.4.6 Asynchronous FIFO Applications 243
18.3 Creating RAM and FIFO Modules for PolarPro and PolarPro II Devices 244
18.3.1 Creating RAM Blocks 244
18.3.1.1 Design Files 246
18.3.2 Creating FIFO Modules 247
18.3.2.1 FIFO Design Files 250
18.3.2.2 FIFO Interface Signals 250
18.3.2.3 FIFO Interface Signal Description 251

Chapter 19: PCI Configuration for 32-bit Parts 255
19.1 Functional Overview 255
19.2 Configuring PCI Registers 256

Chapter 20: PLL/CCM Wizard 261
20.1 Functional Overview 261
20.2 Running PLL/CCM Wizard 262
20.3 Device Family Selection 262
20.4 Setting Up PLLs 263
20.4.1 PLL Configuration 263
20.4.2 CCM Configuration for the PolarPro Device Family 264
20.5 Output Settings 266

Chapter 21: Constraint Manager 267
21.1 Functional Overview 267
21.2 Using Constraint Manager 268
21.2.1 Placement Tab 269
21.2.1.1 Fix Placement Editor 269

© 2008 QuickLogic Corporation

21.2.1.2 Configuration Editor....270
21.2.1.2.1 I/O Configuration for Pinnacle, Pre-Pinnacle and Eclipse Device Families....270
21.2.1.2.2 I/O Configuration for the PolarPro Device Family....272
21.2.1.2.3 Configuring the PolarPro and PolarPro II Devices I/Os using GUI....274
21.2.1.3 Window-Based Placement Editor....280
21.2.2 Timing Tab....281
21.2.2.1 Timing Constraint Editor....281
21.2.2.1.1 Clocks Tab....281
21.2.2.1.2 I/O Ports Tab....281
21.2.2.1.3 Point-to-Point Tab....283
21.2.2.2 Delay Setting....284
21.2.3 Constraint Reader Tab....285
21.2.4 Constraint Manager Toolbar....285

Chapter 22: Design Constraints and Analysis....287

22.1 Design Constraints....287
22.1.1 Timing Constraints....288
22.1.1.1 Clock Signals....289
22.1.1.2 I/O Signals....289
22.1.1.2.1 Net Name Wildcard....290
22.1.1.2.2 Deleting Signals....290
22.1.1.3 Timing Constraints for I/O Ports....290
22.1.1.3.1 Arrival....290
22.1.1.3.2 Setup....290
22.1.1.3.3 Clock-to-Out....291
22.1.1.3.4 Departure Time....291
22.1.1.3.5 Reference Clock....292
22.1.1.3.6 Setup and Arrival....292
22.1.1.3.7 Point-to-Point....292
22.1.1.4 Fan-out Reduction Techniques....294
22.1.1.5 Reducing Set-up Time and Clock-to-Out....294
22.1.2 Placement Constraints....296
22.1.2.1 Placement Editor....296
22.1.2.1.1 Fixing the Placement of GPIO Pads....297
22.1.2.1.2 Fixing the Placement of DDRIO Pads....298
22.2 Drag-and-Drop Functionality....304
22.3 Fix Placement Using the QCF File....306
22.4 Fixing Unused Pins....308
22.4.1 Changing the Sorting Order....309
22.4.2 Panning to an Item....309
22.4.3 Verify Timing Constraints....309
22.5 Highlight Net and Static Timing Analyzer....309
22.6 Highlight a Net....310
22.6.1 Pan to Net Driver....310
22.7 Clock Network Usage....311

© 2008 QuickLogic Corporation

22.7.1 Show Clock Network Usage Tool 311
22.7.2 Using the Show Clock Network Usage Tool 311
22.8 Report File 312
22.8.1 Default Report 312
22.8.1.1 Design Information 312
22.8.1.2 Operating Conditions 313
22.8.1.3 Utilization Information 313
22.8.1.4 Clock Network Utilization by Clock Pads 314
22.8.1.5 Clock Network Utilization by Internal Logic 314
22.8.1.6 Clock Network Utilization by PLL 315
22.8.1.7 Available HSCK/QMUX Clock Networks 315
22.8.1.8 Clock Network Load Information 316
22.8.1.9 Tools Run on Design <design name> 316
22.8.1.10 Pin Table 316
22.8.1.11 IO Banks, Bank-I/Os Info and Their VCCIO, VRef Values 317
22.8.1.12 Fixed Flip Flops 317
22.8.1.13 Fixed RAM Cells 317
22.8.1.14 Fixed ECU Cells 317
22.8.1.15 Nets Removed by Technology Mapper 317
22.8.1.16 Clock Signal Information 317
22.8.1.17 Flip-Flops Clocked by Non-Clock Cells 318

Chapter 23: Static Timing Analyzer 319

23.1 Functional Overview 319
23.2 Using Static Timing Analyzer 320
23.2.1 Static Timing Analyzer Options Pane 321
23.2.2 Selection Pane 322
23.2.2.1 Path Analysis 322
23.2.2.1.1 Path Analyzer Options 323
23.2.2.1.1.1 Path Analysis Information 324
23.2.2.1.1.2 Selecting the Start and Stop Sets 324
23.2.2.1.1.2.1 Select and Move Nets 325
23.2.2.1.1.3 Path Analyzer Scripting 326
23.2.2.1.1.3.1 Record 327
23.2.2.1.1.3.2 Save and Execute 328
23.2.2.1.1.4 Global Clock Domain Path Analysis 328
23.2.2.1.1.5 Processing False and Multi-Cycle Path in Timing Analyzing 331
23.2.2.1.1.6 Graphing 332
23.2.2.1.1.7 Key Calculations 333
23.2.2.1.1.7.1 Clock Skew 334
23.2.2.1.1.7.2 Operating Frequency 334
23.2.2.1.1.7.3 Hold Time 335
23.2.2.1.1.7.4 Clock-to-Output 335
23.2.2.1.2 ESP Core Clock Timings 336
23.2.2.2 Timing Report Summary 336
23.2.2.2.1 Datasheet 337

© 2008 QuickLogic Corporation

23.2.2.2.2 Clock Timing Results 337
23.2.2.2.2.1 Sync --> Sync 338
23.2.2.2.2.2 Pad --> Sync 338
23.2.2.2.2.3 Sync --> Pad 338
23.2.2.2.2.4 Clock --> Sync 339
23.2.2.2.3 Clock Domain Summary 339
23.2.2.2.4 Clock Group Summary 339
23.2.2.2.5 Paths Crossing Clock Boundaries 340
23.2.2.3 Violations 340
23.2.2.3.1 Constraint Violations 341
23.2.2.3.2 Hold Time Violations 341
23.2.2.3.2.1 Clock (Direct and Constrained) 341
23.2.2.3.2.2 Clock Domain 342
23.2.2.3.2.3 Clock Group 343
23.2.3 Viewer Pane 344
23.2.3.1 Saving the List of Generated Paths 345
23.2.3.2 Expanding Paths 345
23.2.4 Static Timing Analyzer Toolbar 347

Chapter 24: OrCAD Schematic 349

24.1 Introduction 349
24.2 OrCAD Project Creation 349
24.3 Creating OLB Files in OrCAD 350

Chapter 25: SpDE Menu Command Reference 351

25.1 What is SpDE? 351
25.2 Viewer 355
25.2.1 Physical Viewer 355
25.2.2 Design Browser 358
25.2.2.1 Filtering the Design Tree 359
25.2.2.2 Design Tree Structure 359
25.2.2.3 Searching in the Design Tree 360
25.2.2.4 Filtering VCC/GND Nets 361
25.2.2.5 Filtering Unused Cells 361
25.2.2.6 Object Properties 362
25.2.2.6.1 Property Structures 362
25.2.3 Package Viewer 365
25.2.4 Viewer Features 365
25.2.4.1 Locating a Cell/Net in the Design Browser 366
25.3 Design Verifier 366
25.3.1 Types of Errors in the Design Verifier 366
25.4 Transcript Window 367
25.4.1 Docking Modes of Transcript Window 368
25.4.2 Transcript Window Tabs 368
25.4.2.1 Status Messages 368

© 2008 QuickLogic Corporation

25.4.2.2 Errors and Warnings 369
25.4.2.3 Verifier Notes 369
25.4.2.4 TCL Console 369
25.4.3 Progress Bar 369
25.5 File Menu 370
25.6 View Menu 371
25.6.1 Texting Tab 372
25.6.2 Color Tab 373
25.6.3 Drawing Tab 374
25.6.4 Windows Tab 375
25.6.5 Tools Tab 376
25.7 Design Menu 377
25.8 Tools Menu 378
25.8.1 Load/Save Tab 379
25.8.2 Migrate Part Tool 380
25.9 Program Menu 381
25.10 Info Menu 382

Chapter 26: Install Devices 385
26.1 Using Install Devices 385

Chapter 27: Web Update 387
27.1 Web Update at Start Up 387
27.2 Using Web Update 388

Appendix A: File Extensions Reference 389
A.1 File Extensions 389

Chapter B: Macro Library 393
B.1 Macro Library Overview 394
B.2 Hard Macros 395
B.3 Soft Macros 395
B.4 I/O Pads 396
B.4.1 CKPAD 397
B.4.2 CKPAD_25um 398
B.4.3 CKPAD2_DYN_EN 398
B.4.4 GCLKBUFF_25um 398
B.4.5 GCLKBUFF 399
B.4.6 QMUX, QHSCKBUFF, QHSCKBUFF2, and QHSCKIBUFF 399
B.4.7 HDPAD 399
B.4.8 HDPAD_25um 400
B.4.9 IO_BUFF_25um 400

© 2008 QuickLogic Corporation

B.4.10 INPAD 400
B.4.11 INPAD_25um 401
B.4.12 OUTPAD 401
B.4.13 OUTPAD_25um 401
B.4.14 TRIPAD 402
B.4.15 TRIPAD_25um 402
B.4.16 BIPAD 402
B.4.17 DDR_DQ 403
B.4.18 BIPAD_25um 403
B.5 Gates 404
B.5.1 Naming Convention 404
B.5.2 AND Gates 404
B.5.3 OR Gates 405
B.5.4 NAND Gates 406
B.5.5 NOR Gates 407
B.5.6 pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro, and PolarPro II-Specific Gates 408
B.5.6.1 DFF 409
B.5.6.2 DFFC 409
B.5.6.3 DFFPC 410
B.5.6.4 DFFE 410
B.5.6.5 DFFEPC 410
B.5.6.6 DFFsC 411
B.5.6.7 TFF 411
B.5.6.8 TFFPC 412
B.5.6.9 TFFE 412
B.5.6.10 TFFEPC 412
B.5.6.11 JKFF 413
B.5.6.12 JKFFPC 413
B.5.6.13 JKnFFPC 414
B.5.6.14 JKNN_FF 414
B.6 Latches 416
B.6.1 Macro Descriptions 416
B.6.1.1 DLA 416
B.6.1.2 DLAC 416
B.6.1.3 DLAE 417
B.6.1.4 DLAEC 417
B.6.1.5 DLAMUX 418
B.6.1.6 DLAEMUX 418
B.6.1.7 DLAD 418
B.6.1.8 DLADINV 419
B.6.1.9 DLAP 419
B.6.1.10 DLADE 420
B.7 Multiplexers 421
B.7.1 Naming Conventions 421
B.7.1.1 MUX2x0 422
B.7.1.2 MUX2Dx0 422

© 2008 QuickLogic Corporation

B.7.1.3 MUX4x0 422
B.7.1.4 MUX2FFx0 423
B.8 Registers 424
B.8.1 Macro Descriptions 424
B.8.1.1 RG4 424
B.8.1.2 RGC4 425
B.8.1.3 RGE4 425
B.8.1.4 RGEC4 426
B.9 0.25 um Device Registers 427
B.9.1 Macro Descriptions 427
B.9.1.1 RG4_25UM 427
B.9.1.2 RGC4_25UM 428
B.9.1.3 RGE4_25UM 428
B.9.1.4 RGEC4_25UM 429
B.9.1.5 DFF_2 429
B.9.1.6 DFFC_2 430
B.9.1.7 DFFE_2 430
B.9.1.8 DFFEPC_2 431
B.10 Shift Registers 432
B.10.1 Macro Descriptions 433
B.10.1.1 SHFT4 433
B.10.1.2 BSHFT4 433
B.10.1.3 LSHFT2Q2 433
B.11 Up Counters 435
B.11.1 QuickLogic Device-Specific Counters 436
B.11.1.1 RCNT4 437
B.11.1.2 UCNTE4 437
B.11.1.3 UCNTL4 437
B.11.1.4 UCNTX4 438
B.11.1.5 UCT8P2 438
B.11.1.6 UCTE16P2 439
B.11.1.7 UCTX16P2 439
B.11.1.8 UPFLCT4 440
B.11.1.9 UPFXCT4 440
B.12 Down Counters 441
B.12.1 Macro Descriptions 442
B.12.1.1 DCNTE4 442
B.12.1.2 DCNTL4 442
B.12.1.3 DCNTX4 442
B.13 Up/Down Counters 444
B.13.1 Macro Descriptions 444
B.13.1.1 UDCNT3 444
B.13.1.2 UDCNT6 445
B.13.1.3 UDCNT12 445
B.13.1.4 UDLF24 446
B.14 pASIC Arithmetic 447
B.14.1 Macro Descriptions 448

© 2008 QuickLogic Corporation

B.14.1.1 HADD1 448
B.14.1.2 FADD1 448
B.14.1.3 FADD1_P2 449
B.14.1.4 RADD4 450
B.14.1.5 ADD4 450
B.14.1.6 FSTADD4 450
B.14.1.7 SUB4 451
B.14.1.8 ACCUM4 451
B.14.1.9 MULT4x4 452
B.14.1.10 COMP4 452
B.14.1.11 ECOMP4 452
B.14.2 QuickLogic Device-Specific Adders and Subtracters 453
B.14.2.1 ADD16P2 453
B.14.2.2 SUB16P2 454
B.15 Combinatorial Macros 455
B.15.1 Macro Descriptions 455
B.15.1.1 XOR2i0 455
B.15.1.2 XOR3i0 455
B.15.1.3 XOR4i0 456
B.15.1.4 XOR5i0 456
B.15.1.5 XNOR2i0 457
B.15.1.6 XNOR3i0 457
B.15.1.7 XNOR4i0 458
B.15.1.8 XNOR5i0 458
B.15.1.9 DEC2t4 458
B.15.1.10 DECE2t4 459
B.15.1.11 MAJ3i0 459
B.16 RAM Blocks 461
B.16.1 Macro Descriptions 461
B.16.1.1 RAM64 x 18 461
B.16.1.2 RAM128 x 9 462
B.16.1.3 RAM256 x 4 463
B.16.1.4 RAM512 x 2 463
B.17 RAM Blocks for 0.25 um Devices 465
B.17.1 Macro Descriptions 465
B.17.1.1 RAM1024 x 2_25UM 465
B.17.1.2 RAM128 x 18_25UM 466
B.17.1.3 RAM256 x 9_25UM 467
B.17.1.4 RAM512 x 4_25UM 467
B.18 RAM Blocks for PolarPro Devices 468
B.19 RAM Blocks for PolarPro II Devices 469
B.20 PCI Interface Cores Macros 470
B.20.1 Macro Descriptions 471
B.20.1.1 PCI32_25 471
B.20.1.2 PCI32TV2 472
B.20.1.3 PCI32V2 473
B.21 Embedded Computational Units (ECUs) 474

© 2008 QuickLogic Corporation

B.21.1 Macro Descriptions 474
B.21.1.1 ACCUM16_25UM 474
B.21.1.2 ACCUM8_25UM 474
B.21.1.3 ADD16_25UM 475
B.21.1.4 ADD16REG_25UM 475
B.21.1.5 ADD16REG_25UM 476
B.21.1.6 ADD8REG_25UM 476
B.21.1.7 MAC16_25UM 477
B.21.1.8 MULT_ADD16_25UM 478
B.21.1.9 MULT_ADD16REG_25UM 479
B.21.1.10 MULT8x8_25UM 480
B.21.1.11 MULT8X8REG_25UM 481
B.22 Low Voltage Differential Signal (LVDS) 482
B.22.1 Macro Descriptions 482
B.22.1.1 LVDS_INPAD_25DC 482
B.22.1.2 LVDS_OUTPAD_25DC 483
B.22.1.3 REG_LVDS_OUTPAD_25DC 483
B.23 Low Voltage Positive Emitter Coupled Logic (LVPECL) 485
B.23.1 Macro Descriptions 485
B.23.1.1 LVPECL_INPAD_25 485
B.23.1.2 LVPECL_OUTPAD_25AC 486
B.23.1.3 LVPECL_OUTPAD_25DC 486
B.23.1.4 REG_LVPECL_OUTPAD_25AC 487
B.23.1.5 REG_LVPECL_OUTPAD_25DC 487
B.24 7400-Series TTL 489
B.25 Master Cells 494
B.26 pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro and PolarPro II-Specific Macros 495
B.27 pASIC3, QuickRAM, and QuickPCI-Specific Macros 497
B.27.1 Hard Macros 497
B.27.1.1 Logic Cell Macro (LOGIC2) 498
B.27.1.2 Internal Global Network Buffer (GCLKBUFF) 498
B.27.1.3 I/O Macros 498
B.27.1.3.1 CKPADff, CKdPADff, and CKtPADff 499
B.27.1.3.2 HDPADff, HDiPADff, and HDdPADff 500
B.27.1.3.3 INPADff 501
B.27.1.3.4 BIPADff, BIiPADff, and BIorPADff 501
B.27.2 Master Cells 502
B.27.2.1 LCELL2 502
B.27.2.2 BICELL2 502
B.27.2.3 INCELL2 503
B.27.2.4 CKCELL2 503
B.28 QuickPCI-Specific Macros 504
B.29 Eclipse-Specific Macros 505
B.29.1 Logic Cell Macro (SUPER_CELL) 508
B.29.2 Master Cells for 0.25um Devices 508
B.29.2.1 SUPER_CELL 509

© 2008 QuickLogic Corporation

B.29.2.2 EIO_CELL 510
B.29.2.3 INCELL_25um 510
B.29.2.4 CKCELL_25um 511
B.30 PolarPro-Specific Macros 512
B.30.1 Logic Cell Macro (logic_cell) 512
B.30.2 Master Cells for PolarPro Devices 512
B.30.2.1 LOGIC_CELL_MACRO 513
B.30.2.2 GPIO_CELL_MACRO 514
B.30.2.3 DDRIO_DQ_CELL_MACRO, DDRIO_DQS_CELL_MACRO 515
B.30.2.4 RAM4K_2X1_CELL_MACRO 516
B.31 PolarPro II-Specific Macros 517
B.31.1 Master Cells for PolarPro II Devices 517
B.31.1.1 LOGIC_CELL_MACRO 518
B.31.1.2 GPIO_CELL_MACRO 519
B.31.1.3 RAM2K_2X1_CELL_MACRO 520
B.31.1.4 RAM4K_2X1_CELL_MACRO 521

Appendix C: QDIF File Format 523

C.1 QDIF Netlist Sections 523
C.1.1 Library and Logic 523
C.1.2 Syntax 523
C.1.3 Identifiers 524
C.1.4 Keywords 524
C.1.5 Supply Nets 524
C.1.6 Object Count 524
C.2 QDIF Example 525

Appendix D: Error Messages 529

D.1 Error Sources 529
D.2 SpDE Design Verifier 530
D.2.1 Notes 530
D.2.2 Warnings 530
D.2.3 Errors 531
D.2.4 Fatal Errors 531
D.3 SpDE Tool Errors 535
D.3.1 XX - (starting with any two letters) 536
D.3.2 DB - SpDE Database Module 536
D.3.3 CR - Clock Router 536
D.3.4 ED - EDIF Netlist Reader 536
D.3.5 LR - Logic Re-Optimizer 536
D.3.6 LS - Load and Save Files 537
D.3.7 NM - Net Lister 537
D.3.8 RT - Router 538
D.3.9 SD - SDF Writer 538

© 2008 QuickLogic Corporation

D.3.10 SQ - Sequencer 538
D.3.11 TP - Placer 538
D.3.12 UL - SpDE CLI 539
D.3.13 VE - SpDE Physical Viewer 539
D.3.14 VG - Verilog Netlister 539
D.3.15 VR - Verifier 539
D.4 Internal Errors 539
D.5 Out of Memory Errors 540
D.6 SpDE PolarPro Design Verifier 541
D.6.1 Warnings 541
D.6.2 Errors 542
D.6.3 Fatal Errors 542
D.7 SpDE PolarPro Tool Verifier 544
D.7.1 Fix Placement 544
D.7.2 Timing Driven Placer 544
D.7.3 RAM ROM Wizard 545
D.7.4 Power Calculator 545
D.7.5 Logic Optimizer 546

Appendix E: Setting Up Third-Party Simulators 547

E.1 VHDL Simulator Functional Overview 547
E.2 Verilog Simulator Functional Overview 547
E.3 ModelSim for PC Platform 548
E.4 Active HDL for PC Platform 549
E.5 Synopsys VSS for Unix Platform 550

Appendix F: QuickWorks Icons 551

F.1 QuickWorks Windows Icon Group 551
F.2 SpDE/Toolbar Menu Icon Group 551

Appendix G: Synthesis Tool Support: Leonardo Spectrum 555

G.1 Functional Overview 555
G.2 Using Leonardo Spectrum 556
G.3 Tcl Script Support 558
G.4 EDIF Design in QuickWorks 559
G.5 Writing HDLs for Leonardo Spectrum Synthesis Tool 559
G.6 Specific Settings to be Selected in Leonardo Spectrum 559

© 2008 QuickLogic Corporation

Appendix H: Precision RTL Synthesis Tool 561
H.1 Functional Overview 561
H.2 Using Precision RTL Synthesis 561
H.3 Tcl Script Support 564
H.4 Loading EDIF in QuickWorks 564

Appendix I: QuickTools: UNIX Version of QuickWorks 567
I.1 Installation Instructions 567
I.1.1 Installation for Sun/Solaris 567
I.1.2 Installation for Red Hat Linux 567
I.2 Installing QuickTools Server 569
I.3 Installing QuickTools Client 572
I.4 System Requirements - Solaris 573
I.4.1 Hardware Requirements 573
I.4.2 Software Requirements 573
I.5 System Requirements - Linux 573
I.5.1 Hardware Requirements 573
I.5.2 Software Requirements 573
I.6 Operating System Patches 575
I.7 Utilities on UNIX Workstation 577

Appendix J: Using ViewDraw with QuickWorks 579
J.1 Functional Overview 579
J.2 Creating New Project Using the Project Manager 580
J.3 Creating a Schematic Design in ViewDraw 582
J.4 Exporting in EDIF 583
J.5 Import EDIF Netlist into SpDE 584

Appendix K: QCF File Constraints 585
K.1 Introduction 585
K.2 Constraint Options 585
K.2.1 I/O Banks Configuration Constraints 587
K.2.1.1 Designating the I/O Bank Standard 587
K.2.1.2 Configuring the I/O Port Interface Standard 588
K.2.1.3 Designating the Set Standard 588
K.2.1.4 Assigning Gates to Banks 589
K.2.1.5 Assigning Gates to Sets 589
K.2.1.6 Designating Slew to Banks 589
K.2.1.7 Designating DriveStrengthP to Banks 589
K.2.1.8 Designating Slew to Sets 589
K.2.1.9 Designating DriveStrengthP to Sets 590
K.2.1.10 Designating DriveStrengthN to Sets 590
K.2.1.11 Setting Pullup 590

© 2008 QuickLogic Corporation

K.2.1.12 Setting Bank Voltage 591
K.2.1.13 Setting Set Voltage 591
K.2.2 I/O Pad Related Constraints 591
K.2.2.1 Fix-Placing I/O Signals to Pins 591
K.2.2.2 Enabling the I/O Pull-Down Resistor 592
K.2.2.3 Setting the Slew Rate for I/O Pads (Eclipse and PolarPro II Devices) 592
K.2.2.4 Tying Unused I/Os to Default Values 593
K.2.2.5 Setting the Fixhold for I/O Pads (PolarPro II) 593
K.2.3 Timing Related Constraints 593
K.2.3.1 Frequency, Setup, and Clock-to-Out Constraints 593
K.2.3.2 Setting Path Timing Constraints 593
K.2.3.3 Setting Arrival Times for Incoming Signals 594
K.2.3.4 Setting Departure Times for Outgoing Signals 594
K.2.3.5 Setting Clock-to-Out Delays from a Flip-Flop to an Output Pin 595
K.2.3.6 Placing SetupTime Constraints for an Incoming Signal to a Flip Flop 595
K.2.3.7 Setting False Paths—Nets Not Included in Frequency Calculation 596
K.2.3.8 Setting Multi-Cycle Paths—Nets with Multi-Cycle Propagation Delay 597
K.2.3.9 Setting Point-to-Point Timing Constraints 598
K.2.4 Buffering Techniques 598
K.2.4.1 Split Buffering 599
K.2.4.2 Selective Buffering 599
K.2.4.3 Duplicating Gates 600
K.2.5 Buffering/Fanout-Related Constraints 600
K.2.5.1 Prohibiting Buffering on Nets 600
K.2.5.2 Duplicating Drivers with High Fanout 601
K.2.5.3 Turning Off the Duplication Feature 601
K.2.5.4 Setting Fanout Limits on All Nets 601
K.2.5.5 Duplicating Nets 602
K.2.5.6 Duplicating Specific Nets Using a Global Fanout Limit 602
K.2.5.7 Duplicating Specific Nets Using a Specific Fanout Limit 602
K.2.6 Flip-Flops, RAM Blocks, ECU Blocks, and Logic Modules Fix Placement Constraints 602
K.2.6.1 Fixing Flip-Flop Placement 602
K.2.6.2 Pulling ALL Flip-Flops Into I/O Cells 603
K.2.6.3 Pulling a Flip-Flop Into an I/O Cell 603
K.2.6.4 Prohibiting the Use of I/O Flip-Flops 603
K.2.6.5 Fix-Placing ECU Blocks 603
K.2.6.6 Fix-Placing RAM Blocks 603
K.2.6.7 Fix-Placing Logic Modules 604
K.2.6.8 Manual Clock Assignment 604
K.2.6.9 Clock Enable Assignment 605
K.2.6.10 CCM Placement 605
K.3 Conclusion 605
Index 607

© 2008 QuickLogic Corporation

Chapter 1
Introduction

The QuickWorks® software package from QuickLogic offers a comprehensive and easy-to-use development toolset for QuickLogic pASIC® and FPGA (Field Programmable Gate Arrays). It provides a complete cycle of operations, from design entry to creation of program file, for Windows-based PC platforms. QuickTools is the UNIX version of QuickWorks.
For more information on QuickTools, refer to **Appendix I, "QuickTools: UNIX Version of QuickWorks" on page 567**.

This chapter includes the following topics:

- **"Product Information" on page 1**
- **"Package Contents" on page 2**
- **"About This Manual" on page 2**
- **"Technical Support" on page 3**

1.1 Product Information

The QuickWorks package offers the following software components:

- Basic QuickWorks
  - Schematic Capture
  - Timing-Driven Automatic Place and Route
  - Timing Analysis
  - Interfaces to Third Party EDA Tools
  - Support for Windows® 98, 2000, Me, NT 4.0, XP
  - Waveform Simulation
- Aldec Active-HDL Verilog/VHDL Simulator
  - Verilog Simulation
  - VHDL Simulation
- Precision RTL Synthesis
  - VHDL Synthesis
  - Verilog Synthesis

Using QuickWorks, designers can mix a combination of schematic and Verilog/VHDL using the schematic design entry tool. Symbols within the schematic editor can reference VHDL or Verilog files allowing a mixture of schematic design and HDL design. The designer can then simulate functionally (zero-delay), run place and route, and then simulate with full timing information. The static timing analyzer, timing driven placement, and other features (design optimizer and compiler) are also available to aid in the design process.

A 30-day Evaluation Kit of QuickWorks is available, free of charge from QuickLogic. The Evaluation Kit comes with documentation and supports all QuickWorks features, allowing you to completely evaluate your design from start to finish.

© 2008 QuickLogic Corporation

NOTE: For UNIX version, refer to **Appendix I, "QuickTools: UNIX Version of QuickWorks" on page 567**.

1.2 Package Contents

The QuickWorks Suite comes in Ala Carte Packages, as well as Bundled packages, to enable you to choose the components required for your design. It contains the following items:

- Basic QuickWorks
  - QuickWorks Development Tools CD-ROM
  - QuickLogic Programming Tools CD-ROM
    - Manuals
    - QuickWorks Users Guide
    - Programmer Kit User's Guide
    - Schematic Editor User's Guide
    - Schematic Editor Reference Guide
    - QuickLogic Hardware Key
    - QuickLogic License Agreement
    - QuickWorks Authorization Instructions Letter
    - Welcome to QuickWorks Letter
- Aldec Active-HDL Verilog Simulator
  - Aldec Active-HDL VHDL Simulator
  - QuickWorks Development Tools CD-ROM
  - Manuals
    - QuickWorks VHDL Simulation License Instructions Letter

1.3 About This Manual

The *QuickWorks User Manual* is separated into sections according to product usage:

- **Getting Started**—Read this section to install and license your QuickWorks software.
- **Overall Design Process**—Use this section for an overview of the complete design process.
- **Development Tools**—This section is organized by the tasks used to implement a design.
- **SpDE Command Reference**—This section is organized by the commands listed in the SpDE menu.
- **Macros**—This section contains detailed information on each QuickLogic schematic macro.
- **Appendices**—This section includes pin-out tables, diagrams, and error message descriptions.

© 2008 QuickLogic Corporation

1.4 Technical Support

If you have any questions that are not addressed in the existing documentation, please contact the QuickLogic technical support team at:

USA

Telephone: (408) 990-4100

Fax: (408) 990-4040

E-mail: info@quicklogic.com

Support: http://www.quicklogic.com/support

Internet: http://www.quicklogic.com

Europe

Telephone: + (44) 1932-57-9011

Fax: + (44) 1815-63-9254

© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation

Chapter 2
Requirements and Installation

This chapter includes the following topics:

- **“Hardware Requirements” on page 5**
- **“Software Requirements” on page 5**
- **“QuickWorks Installation” on page 6**

NOTE: For information on Requirements and Installation of QuickTools, refer to **Appendix I, “QuickTools: UNIX Version of QuickWorks” on page 567**.

2.1 Hardware Requirements

- Pentium III 500 MHz or greater recommended
- Serial port (for programming devices)
- Parallel port (for the security key)
- 512 MB of RAM
- At least 512 MB Virtual Memory (Windows Permanent Swap File)
- At least 2 GB of free hard disk space

2.2 Software Requirements

You can install the QuickWorks tool set on any of the following operating systems:

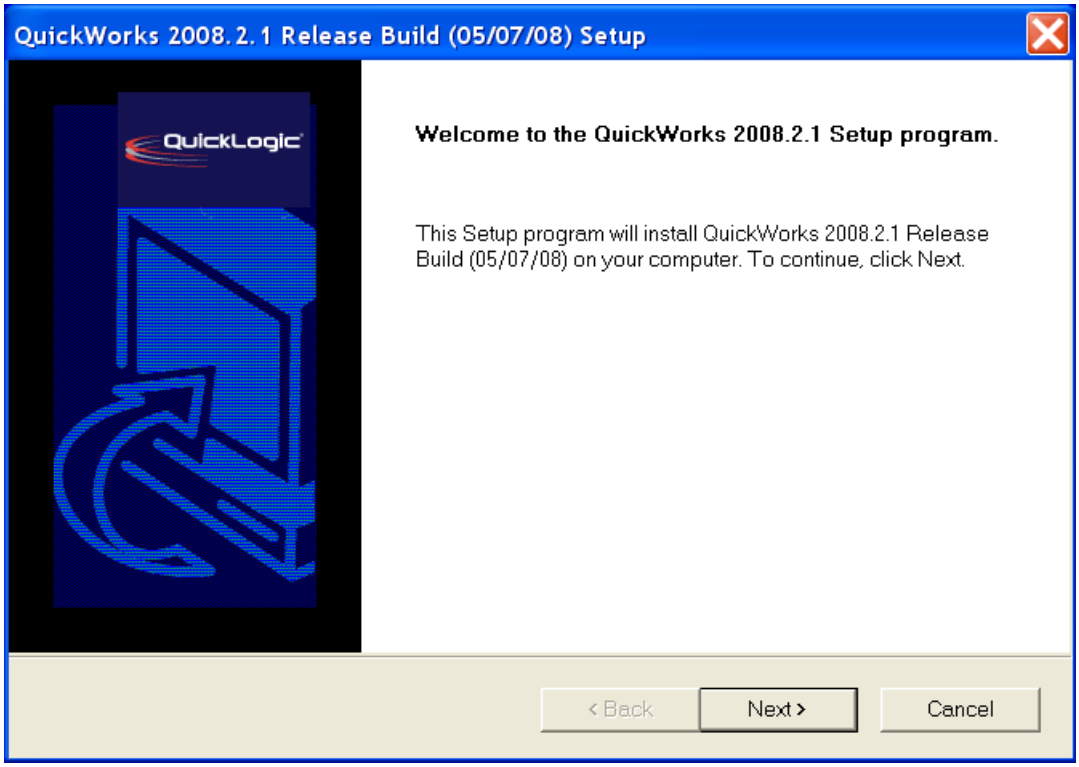
- Microsoft® Windows® NT, 2000, XP, or Vista
- Microsoft Access™ or MS Office

© 2008 QuickLogic Corporation

2.3 QuickWorks Installation

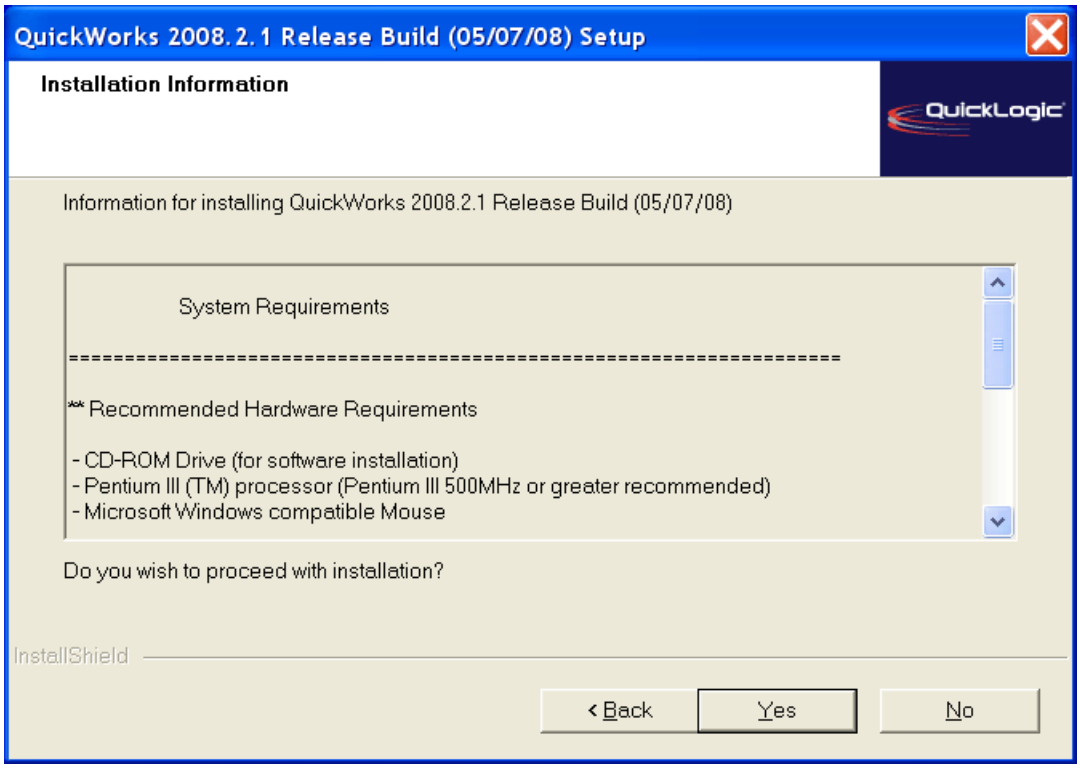
NOTE: For installation on UNIX, refer to **Appendix I, "QuickTools: UNIX Version of QuickWorks" on page 567**.

1. Insert the QuickWorks CD into your CD-ROM drive. The QuickLogic Installation dialog box displays automatically.
2. Click **QuickWorks** to start the QuickWorks installer.
3. The Welcome screen is displayed, click **Next**.

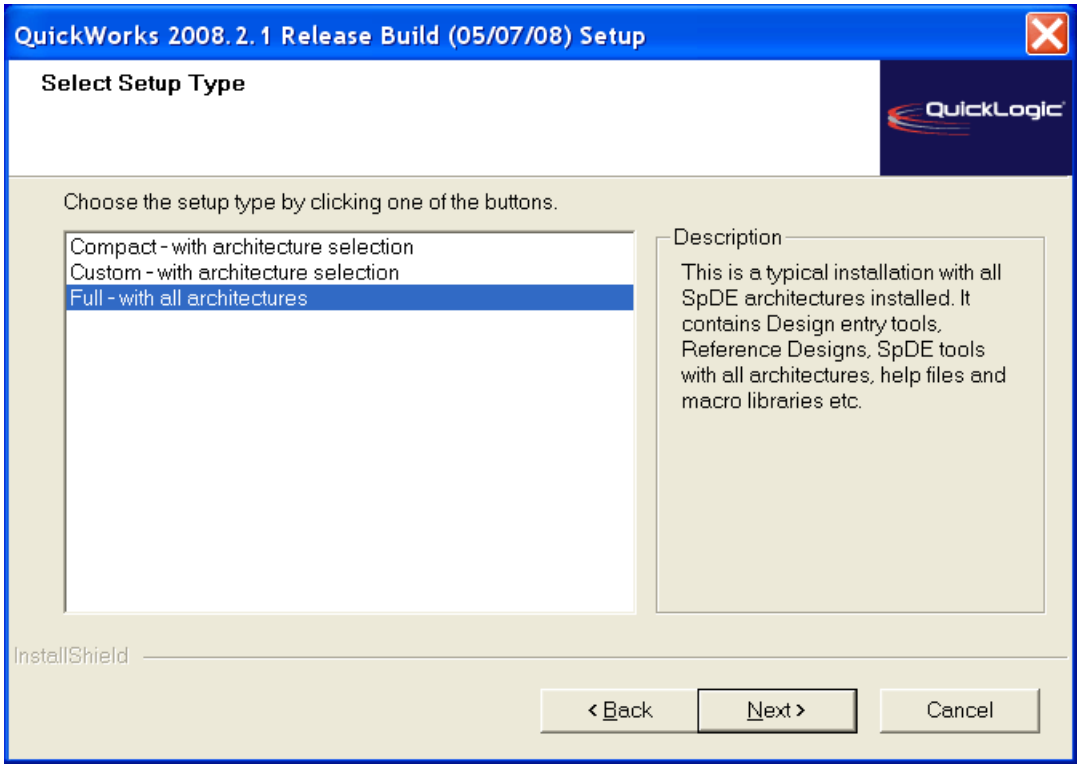

QuickWorks 2008.2.1 Release Build (05/07/08) Setup
QuickLogic
Welcome to the QuickWorks 2008.2.1 Setup program.
This Setup program will install QuickWorks 2008.2.1 Release Build (05/07/08) on your computer. To continue, click Next.
< Back
Next >
Cancel

© 2008 QuickLogic Corporation

4. The system requirements are displayed. Read the installation information to decide if you want to proceed with the installation of QuickWorks on your machine. Click **Yes** to continue installation or click **No** to quit.


QuickWorks 2008.2.1 Release Build (05/07/08) Setup
Installation Information
QuickLogic
Information for installing QuickWorks 2008.2.1 Release Build (05/07/08)
System Requirements
** Recommended Hardware Requirements
- CD-ROM Drive (for software installation)
- Pentium III (TM) processor (Pentium III 500MHz or greater recommended)
- Microsoft Windows compatible Mouse
Do you wish to proceed with installation?
InstallShield
< Back
Yes
No

5. In the Select Setup Type dialog box, choose your preferred setup type and click **Next**.


QuickWorks 2008.2.1 Release Build (05/07/08) Setup
Select Setup Type
QuickLogic
Choose the setup type by clicking one of the buttons.
Compact - with architecture selection
Custom - with architecture selection
Full - with all architectures
Description
This is a typical installation with all SpDE architectures installed. It contains Design entry tools, Reference Designs, SpDE tools with all architectures, help files and macro libraries etc.
InstallShield
< Back
Next >
Cancel

Compact architecture—installs the Design Entry, schematic Macro Libraries and SpDE Tools. You can choose to install the desired architectures with SpDE Tools.

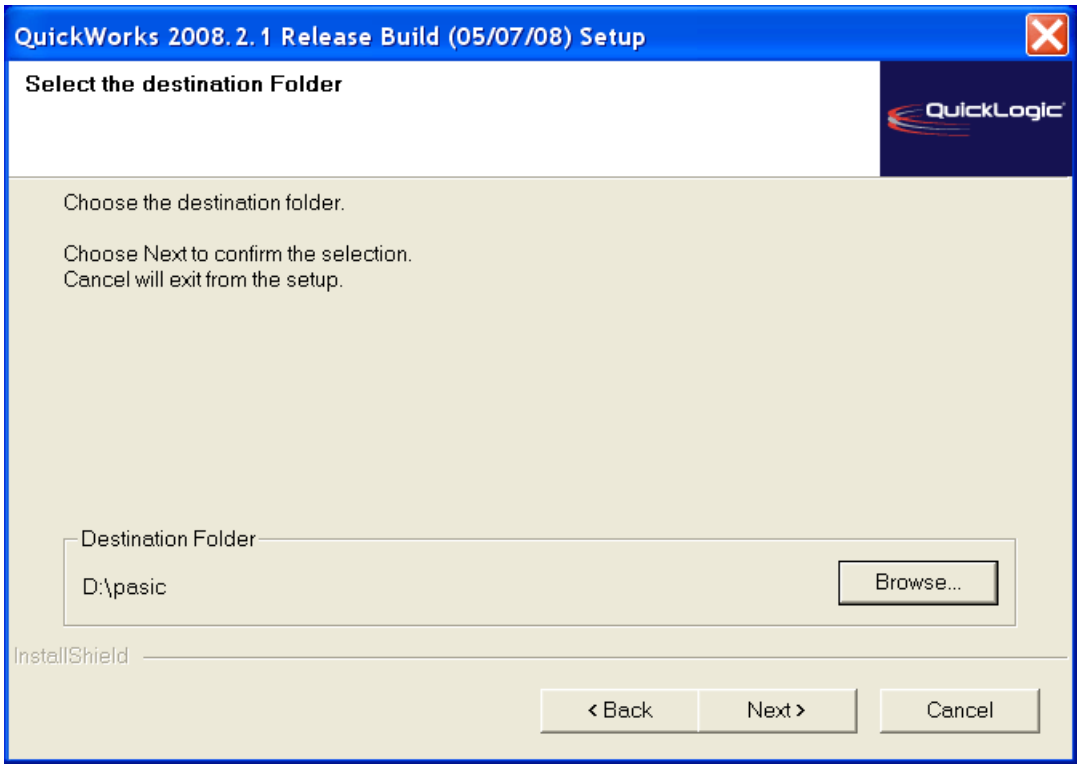
Custom architecture—installs the SpDE Tools and allows selective installation of Design Entry tools plus desired architectures with SpDE Tools, SpDE Help Files, Macro Libraries, etc.

Full with all architectures—typical installation with all SpDE architectures installed. It contains Design Entry tools, SpDE Tools with all architectures, SpDE Help Files, Macro Libraries, etc.

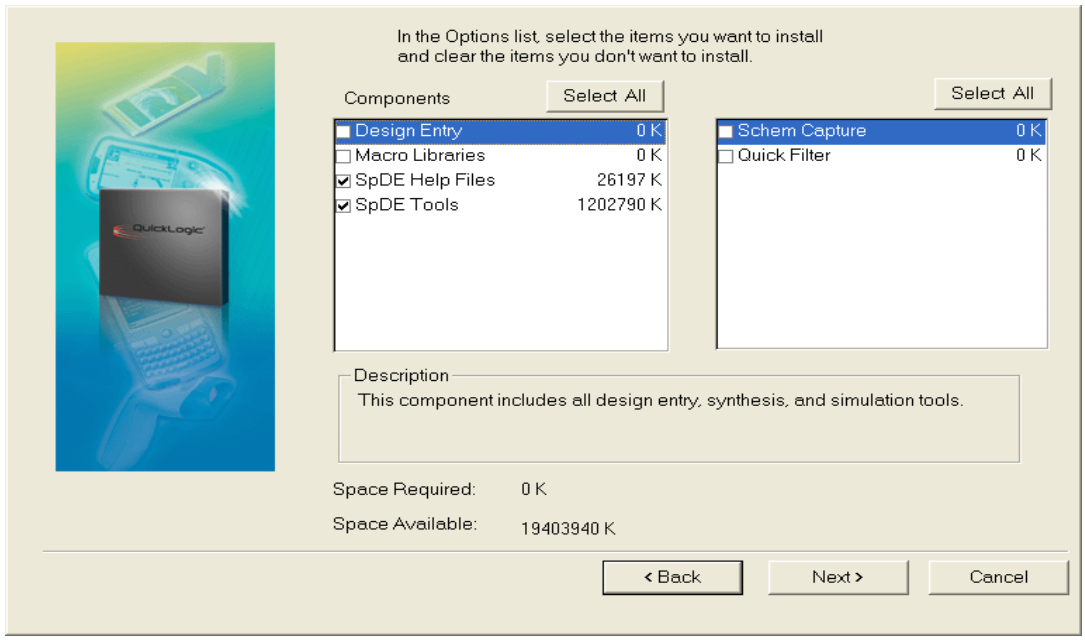
NOTE: If you selected *Full - with all architectures*, proceed to step 7.
If you selected *Custom - with architecture selection,* proceed to step 8.

© 2008 QuickLogic Corporation

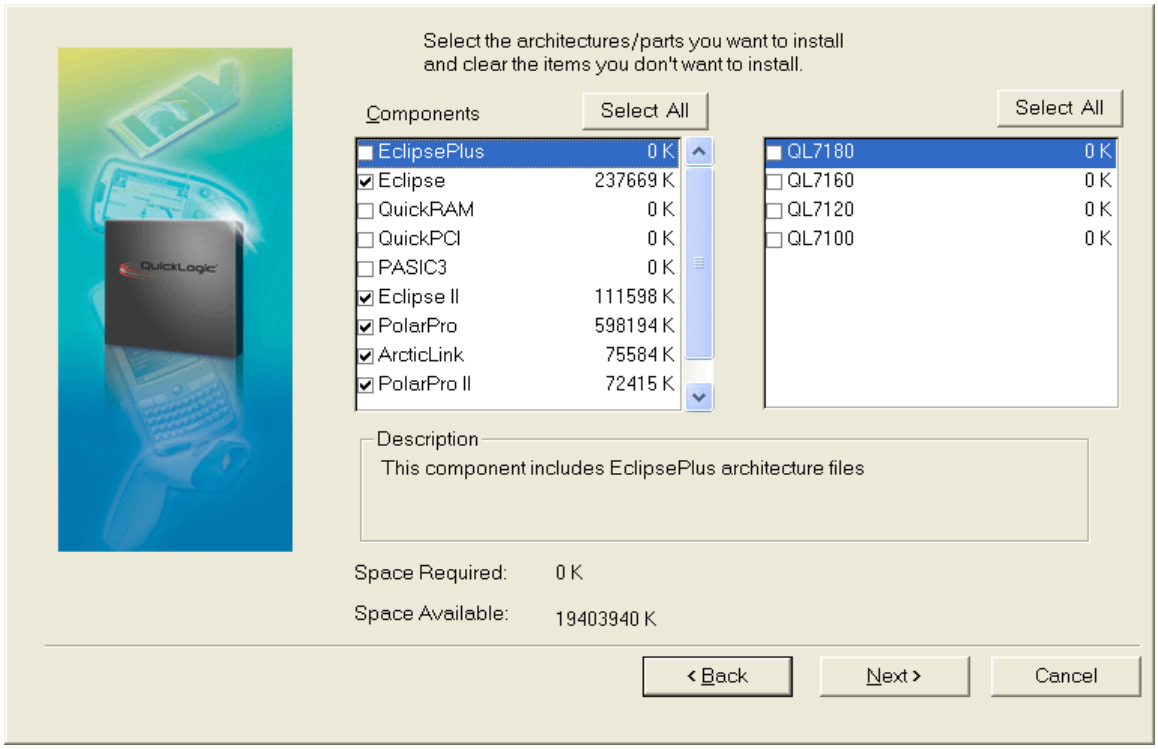
6. In the Select the Destination Folder dialog box, click **Browse** to change the installation directory if you want to use a different directory than the default, shown in the Destination Folder field, and then click **Next**.



7. In the Custom setup dialog box, choose the components you want to install and click **Next**.

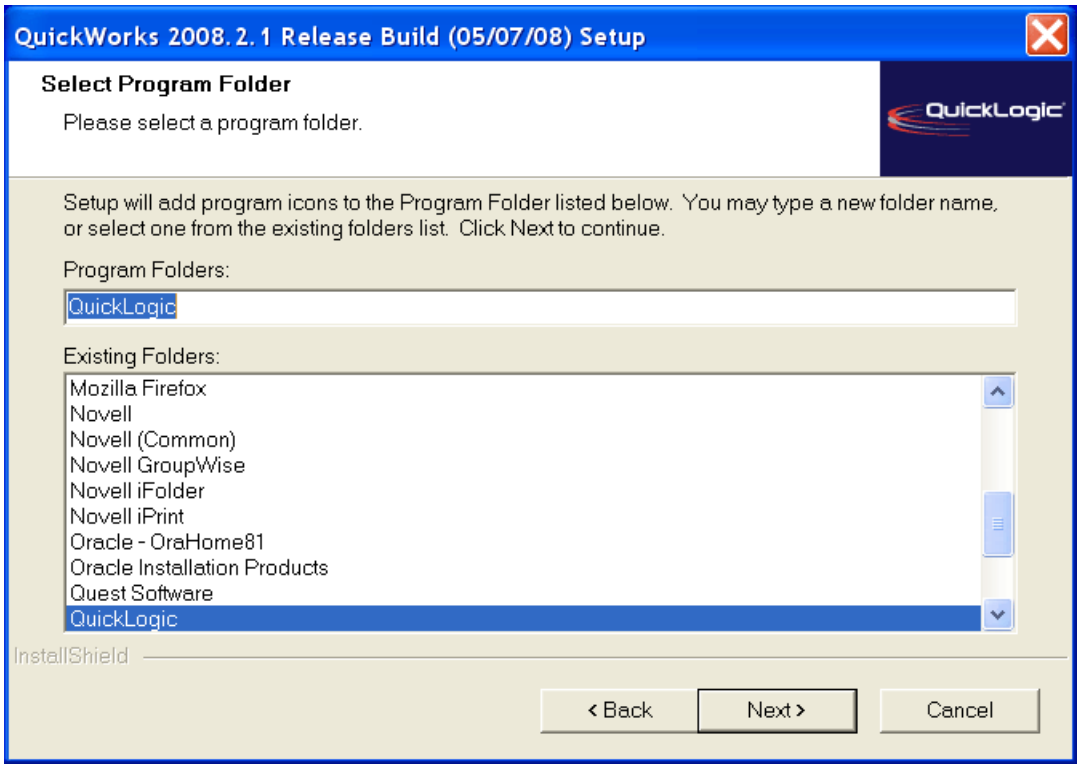

In the Options list, select the items you want to install
and clear the items you don't want to install.
Components
Select All
Design Entry 0 K
Macro Libraries 0 K
SpDE Help Files 26197 K
SpDE Tools 1202790 K
Select All
Schem Capture 0 K
Quick Filter 0 K
Description
This component includes all design entry, synthesis, and simulation tools.
Space Required: 0 K
Space Available: 19403940 K
< Back
Next >
Cancel

8. Compact and Custom setup let you select the desired architectures/parts to install. Eclipse, Eclipse II, and Eclipse-E are selected by default but you can deselect these options if you do not want to install them. Choose the architectures/parts you want to install and click **Next**.

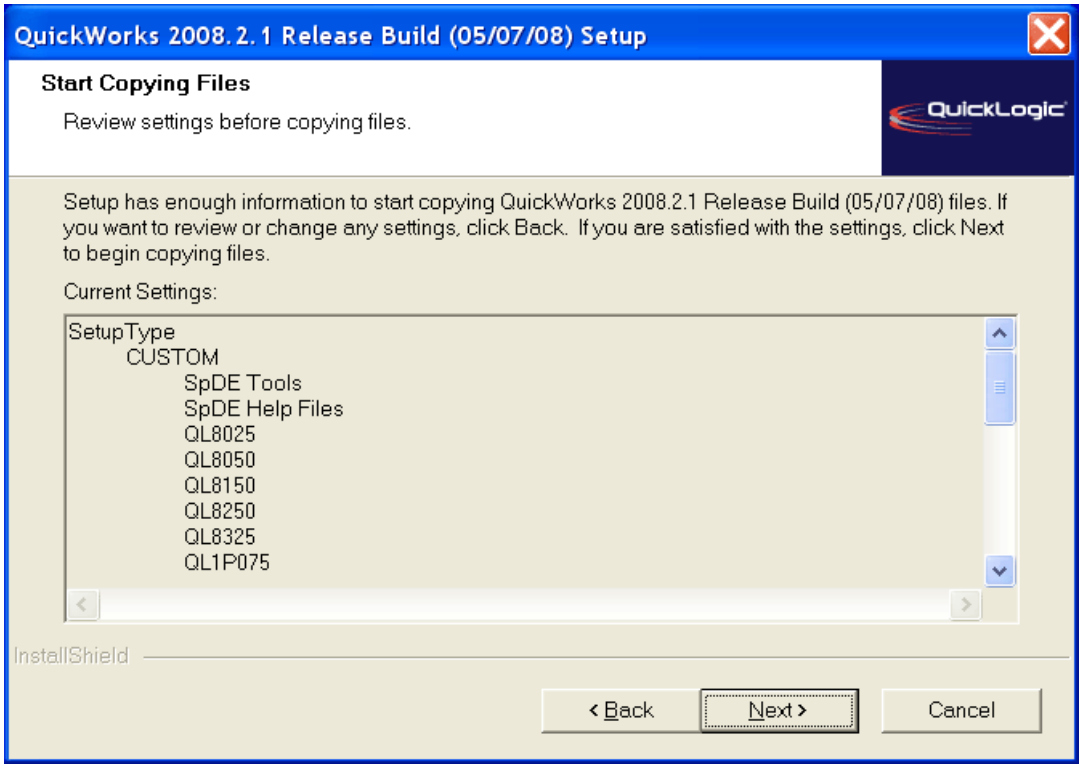


© 2008 QuickLogic Corporation

9. In the Select Program Folder dialog box, choose the destination program folder for the QuickWorks files (default is *QuickLogic*) and then click **Next**.



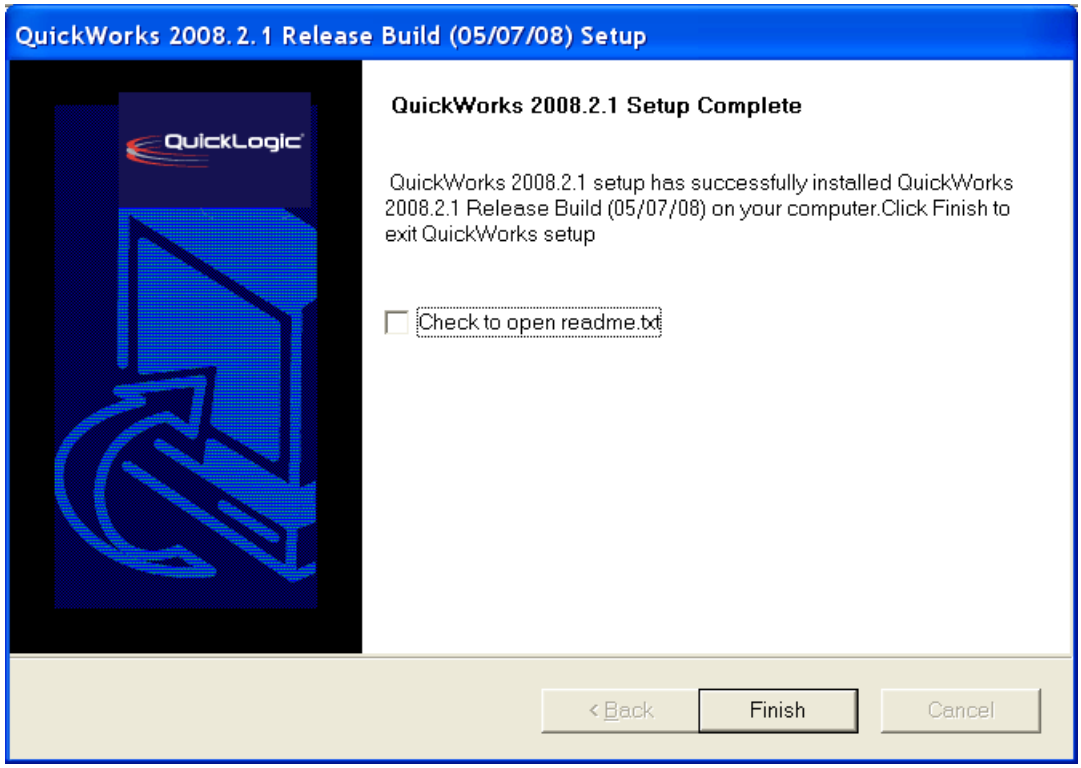
10. In the Start Copying Files dialog box click **Next** to start copying the files. (If you want to change certain settings click **Back**.)



After finishing copying the files, the QuickWorks Setup program automatically updates your PATH variable on all supported operating systems except Windows 98/Me. For Windows 98/Me, the installer informs you that your PATH variable needs to be updated in the AUTOEXEC.BAT. You can either allow the installer to proceed, or you can edit the AUTOEXEC.BAT manually to add the following to the PATH environment variable: `DRIVENAME:\pasic\spde\directory` (where *directory* is the QuickWorks home directory).

NOTE: You must have administrator privileges to be allowed to change the PATH variable.

11. Once the setup has completed, a Setup Complete dialog box opens informing you that the installation was successful.
Click **Finish** to end the setup process for QuickWorks installation.


QuickWorks 2008.2.1 Release Build (05/07/08) Setup
QuickLogic
QuickWorks 2008.2.1 Setup Complete
QuickWorks 2008.2.1 setup has successfully installed QuickWorks 2008.2.1 Release Build (05/07/08) on your computer.Click Finish to exit QuickWorks setup
Check to open readme.txt
< Back
Finish
Cancel

12. A pop-up message displays informing you that a new QuickLogic synthesis tool is available. Click **OK** to close the display. The installation is complete.

© 2008 QuickLogic Corporation

Chapter 3
Design Overview

This chapter provides an overview of the following:

- **"SpDE Definitions" on page 13**
- **"Design Flows" on page 14**

3.1 SpDE Definitions

Seamless pASIC Design Environment (SpDE) is the framework in which all QuickLogic design flows are managed. From within SpDE, you can launch all QuickWorks tools in order to design, evaluate and fully complete your FPGA design project.

3.1.1 Key Design Concepts

To begin a Programmable Application Specific Integrated Circuit (pASIC) design:

1. Capture or enter the basic logic into the design flow by using either a schematic editor or a text editor for HDL entry. In this first stage, a netlist is created for future simulation, and place-and-route.
2. Perform a functional simulation to verify syntax and functionality. This is called a *prelayout simulation*, as the timing is not yet entered.
3. To save debugging time in the later stages of the design review the simulation results at this stage
4. Once the design is verified, you can select Part-and-Package and you may specify the pinout.
5. Load the design netlist into the design environment. The logic optimization tool uses mapping algorithms to efficiently partition logic into cells. When a design has fanout nets, the automatic buffering tool allows the optimizer to utilize un-used logic cells to buffer the high fanouts.
6. To perform Placement and Routing place the design into the core array of the chip. This placement has a large effect on the eventual timing of the chip architecture. Routing is the process of interconnecting the placed gates.
7. When Placement and Routing are complete, QuickWorks can then compute accurate net delays for back annotation to the original design.
8. To verify the design functionality and timing information perform the Post Layout or Timing Simulation.
9. Once you have verified the design you can start programming

© 2008 QuickLogic Corporation

3.2 Design Flows

This section provides an overview of the four standard design modes:

- Schematic-based designs
- Mixed-Mode designs
- Verilog-only designs
- VHDL-only designs

For more detailed step-by-step instructions, see the *Schematic Design Tutorial* chapter of the *QuickWorks Tutorials User Manual*.

3.2.1 Creating a Schematic-Based Design Flow

QuickLogic's QuickWorks design environment fully supports schematic-based designs. Schematic-based designs typically use macros such as simple logic gates, or more complex macros such as adders or counters. These and other macros are listed in **Chapter B, "Macro Library" on page 393** chapter of this manual.

Figure 3-1: Schematic Entry Design Flow

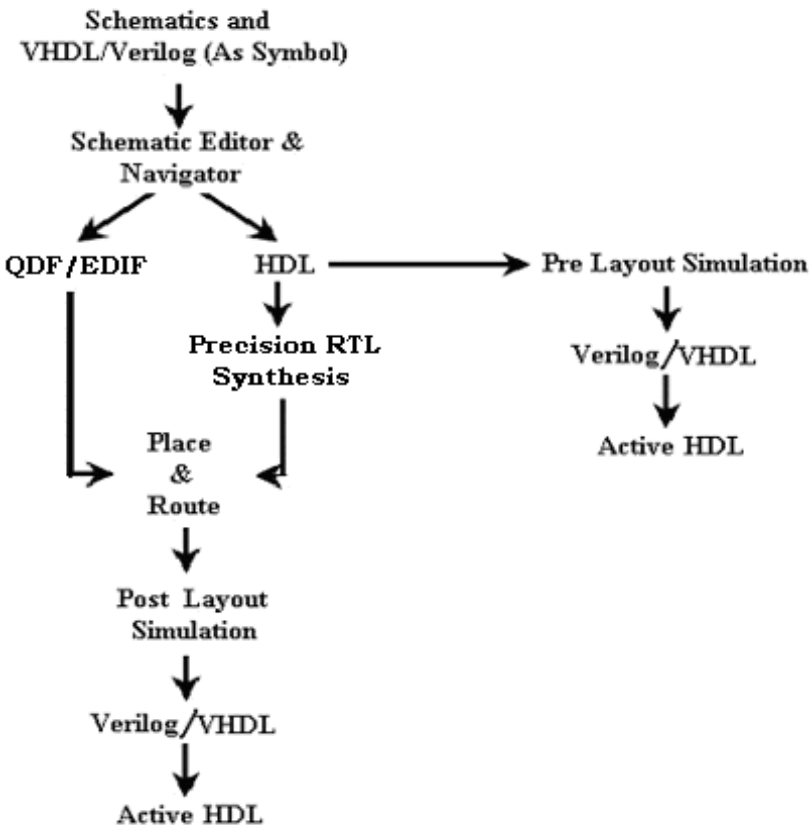


Figure 3-1 illustrates a common schematic-based design flow. The procedure for creating a schematic-based design flow is described in the steps below.

© 2008 QuickLogic Corporation

3.2.1.1 Entering a Design in the Schematic Editor

The Schematic Editor is used to capture and create schematics in the design. Although multi-sheet schematics can be created, this is generally discouraged as they indicate a flat design style rather than a hierarchical design style. SCS allows for easy navigation through a hierarchical design. Once in the Schematic Editor, you can:

- Add symbols and wires
- Add instance names to all symbols (optional). Iterated instances allow a single instance of a symbol to represent many identical instances in a parallel connection. This feature is useful when connecting an array of identical macros to a bus.
- Add net names. Power and Ground Connections are made in the Schematic Editor by naming the net to VCC or GND.
- Add I/O markers to input and output nets
- Run the schematic checker
- Create a symbol for the schematic if it is not a top level design
- Save the design.

3.2.1.2 Building an In-Memory Database with the Hierarchy Navigator

Use the Hierarchy Navigator to build an in-memory database of the completed design, and access the other tools with an interface to this database. Within the Navigator, considered the heart of the QuickWorks schematic design environment, you can perform the following:

- Examine the complete design hierarchy—The Navigator can traverse the design from the broad, top-level schematic where both schematic verilog or VHDL blocks are shown, down to the primitive cells which are the design's basic building blocks. From the Navigator, you may also view detailed information on instances, pins and nets within the design hierarchy.
- Assign attributes within the design hierarchy—While examining the design, attributes may be assigned for fixed placement of cells and for other purposes. These attributes are assigned hierarchically—each attribute is assigned to a unique instance of a component within the design.
- Create and Export a Verilog, VHDL or QDIF/EDIF Netlist—Netlists are a part of the prelayout (before place and route) preparation to run simulation. There are three types of Netlists to create from the Hierarchy Navigator: Verilog, VHDL, and QDIF/EDIF netlist. The QDIF/EDIF or QuickLogic Data Interchange Format may be exported if the design contains no Verilog or VHDL blocks (i.e., schematic-only based designs). However, a Verilog or VHDL netlist must be created for functional simulation.
- Based on the HDL that you will be using for simulation, you need to export a Verilog or VHDL netlist from the Hierarchy Navigator. Once the Netlist is created and exported, you have a structural verilog or VHDL netlist for your top-level design. You can then describe the stimulus for the Simulator either automatically by using the Waveform Editor, or by hand using the Active HDL Simulator.
- Use the Waveform Editor to describe input stimulus graphically and automatically to create a Verilog test fixture. A Verilog test fixture can also be written by hand with the Turbo Writer Editor.

© 2008 QuickLogic Corporation

3.2.1.3 Using SpDE

The SpDE automatic tools will optimize, place, route, sequence, and delay-model the design. For example, the Path Analyzer offers flexible configuration which allows the listing to be filtered to the paths of interest only. The Delay Modeler will generate precise postlayout delays, which will be back-annotated for simulation.

3.2.1.4 Programming a Device

After entering your design and performing the simulation you can start programming a device.

3.2.2 Creating a Mixed-Mode Design Flow

QuickWorks provides the capability to work in a mixed-mode environment with designs containing a mixture of schematics and Verilog or VHDL blocks.

At the first stage of a design entry, the schematic is captured using the Schematic Editor just as it would be for any schematic-only based design. The Verilog or VHDL design is entered using the Turbo Writer Editor. A symbol can be created with the same name as the Verilog or VHDL file to include the HDL file in the schematic hierachy.

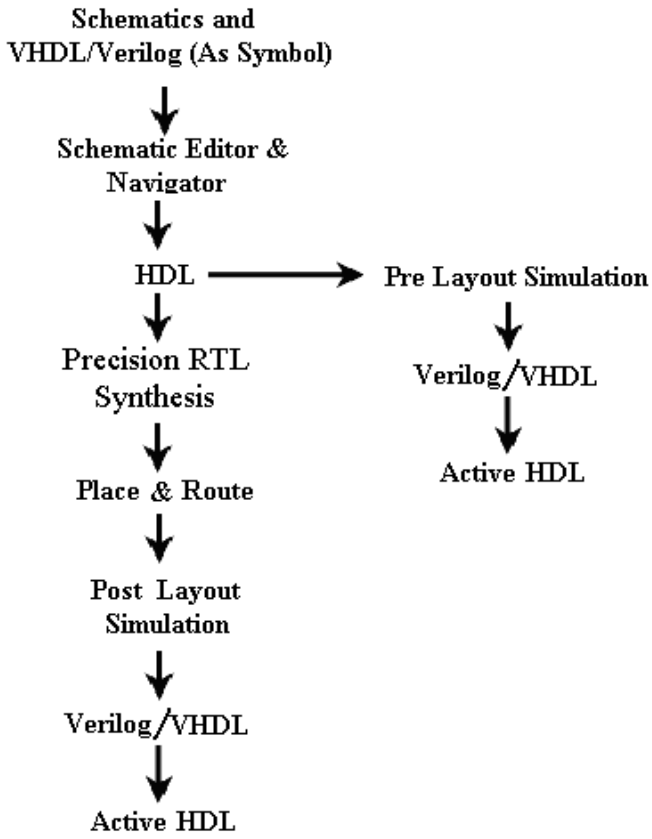
In this way, the QuickWorks toolkit integrates the SCS schematic environment tools with the Turbo Writer context sensitive editor and Precision RTL Synthesis tool so that Verilog or VHDL blocks can be implemented as blocks in a schematic.

Figure 3-2 illustrates the flow of mixed-mode designs. The individual steps are explained in the following section.

© 2008 QuickLogic Corporation

NOTE: You cannot have both VHDL and Verilog blocks in the same design.

Figure 3-2: Mixed-Mode Design Flow



3.2.2.1 Capturing a Design in SCS

The **SCS Schematic Editor** is used to capture the schematic design. The following procedures apply just as for any top-level schematic:

- Add symbols and wires
- Add instance names to all symbols (optional)
- Add net names
- Add I/O markers to input and output nets
- Run the schematic checker, fix any errors
- Save the schematic
- If this is not the top level schematic, create a symbol for the schematic.

The **Turbo Writer** is used to enter any HDL file (Verilog, or VHDL) into the design. Turbo Writer offers the following features:

- **Text editor emulation**—allows emulation from a variety of standard text editing platforms: for example: `vi, brief, epsilon`.
- **Context sensitive editing**
- **Verilog and VHDL language templates**—provides shorthand entry for commonly used descriptions

© 2008 QuickLogic Corporation

- **Color-coded Text**—allows for easy reading and quick comprehension
- **Syntax checking**—The Turbo Writer's Syntax Check verifies the syntax for HDL files. Once the syntax is correct, you return to the Schematic Editor to complete a Top-Level schematic.

The **Schematic Editor** is used to add symbols for both schematic and HDL blocks.

After adding symbols for all design blocks, the consistency is checked once again, looking for mismatches between I/O markers in the schematic designs, and the pins on the corresponding symbols. The consistency checker also finds wiring errors in the schematics.

Once checked and saved, the mixed design is automatically entered into the Hierarchy Navigator.

3.2.2.2 Building an In-Memory Database with the Hierarchy Navigator

All steps in the Hierarchy Navigator are the same for both mixed-mode and schematic-based designs, *except*:

- There are two types of netlists generated for mixed-mode designs:
  - **Verilog netlist** This netlist may be exported if the design does not contain VHDL blocks. If there is any Verilog code in your design, a Verilog netlist is required.
  - **VHDL netlist** If there is any VHDL code in your design, you will need to export a VHDL netlist.
- Once the Netlist is created and exported, you will have a structural Verilog or VHDL netlist for your top-level design which *also contains a reference to the lower-level HDL block*. You then enter stimulus for the Active HDL Simulator from the Waveform Editor.

3.2.2.3 Using the Waveform Editor, Active HDL Simulator, and the Data Analyzer

All the processes of the Waveform Editor, the Active HDL Simulator, and the Data Analyzer are the same for both mixed-mode and schematic-based designs, *except* that once you have viewed the results and the Verilog Netlist file is loaded into SpDE, the file will first be synthesized in Precision RTL.

3.2.2.4 Using Precision RTL Synthesis

Precision RTL is the synthesis product supplied with QuickWorks. It can take a Verilog or VHDL design and target it to a QuickLogic device.

After the compile and synthesis process, a design is complete and will be automatically imported into SpDE. As a result of importing the Verilog netlist through Precision RTL, you now have an EDIF file (`.edf` extension). This edif file can be used to later import your design into SpDE.

3.2.2.5 Using SpDE, Active HDL Simulator, and the Data Analyzer

All the steps described in the Schematic-Based Design are the same for Mixed Schematic-Based Design.

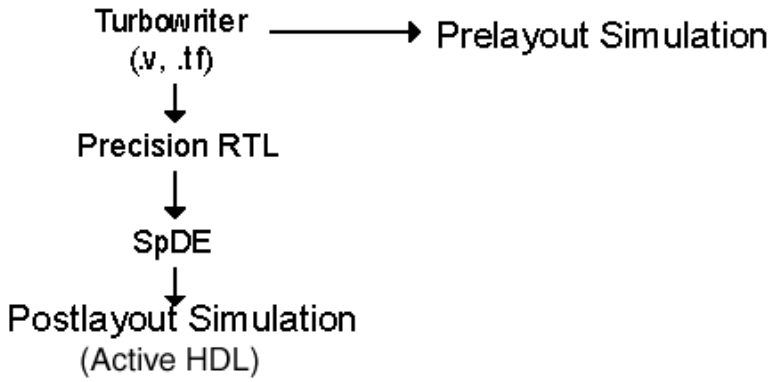
© 2008 QuickLogic Corporation

3.2.3 Creating a Verilog-Only Design Flow

With the evolution of today's design methodology, synthesis is becoming ever more important. The QuickWorks toolkit allows you to describe a complete design at various levels of abstraction from structural to behavioral.

Figure 3-3 on page 19 illustrates the design flow described in the Turbo Writer editor, compiled by Precision RTL during the SpDE import of the design, and placed and routed by the SpDE tools.

Figure 3-3: Verilog-Only Design Flow



3.2.3.1 Entering a HDL Design

To enter a HDL design:

1. Generate Verilog source code in Turbo Writer.
2. Verify the syntax using Turbo Writer's Syntax Check.
3. Create Input Vectors - Verilog Test-Fixture—From within Turbo Writer, a Verilog test-fixture is created either by hand or with the graphical Waveform Editor.

3.2.3.2 Performing the Prelayout Simulation

To perform a prelayout simulation:

1. Functionally simulate the Verilog design.
2. The simulation results may be viewed using the Data Analyzer.
3. The Verilog file will then be exported to SpDE.

3.2.3.3 Performing Synthesize, Place-and-Route

1. Synthesize the Verilog design in Precision RTL.
2. When the Precision RTL compilation is complete, a netlist is created and automatically loaded into SpDE.
3. The design is then automatically optimized, placed, and routed. The Delay Modeler generates precise postlayout delays, which are back-annotated for simulation.

© 2008 QuickLogic Corporation

3.2.3.4 Performing a Postlayout Simulation

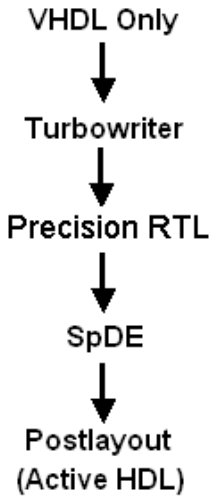
- Simulate the postlayout Verilog netlist.
- View the postlayout simulation results in the Data Analyzer.

3.2.4 Creating a VHDL-Only Design Flow

For VHDL design entry and synthesis, Precision RTL provides a fast and easy-to-use interface with quality synthesis results. QuickWorks provides access to the Active HDL Simulator. QuickWorks also supports popular VHDL simulators such as ModelTech V System, Synopsis VSS, and other VITAL VHDL compliant simulators by exporting a VITAL VHDL netlist.

Figure 3-4 on page 20 illustrates the flow of VHDL designs.

Figure 3-4: Recommended VHDL-Only Design Flow



3.2.4.1 Entering a VHDL Design

1. Enter your VHDL source code in Turbo Writer.
2. Perform syntax check and save the completed code.
3. Create Input Vectors - VHDL Test-Bench by creating a VHDL test bench for the design.

3.2.4.2 Performing a Prelayout Simulation

1. Launch and perform functional/behavioral simulation using either the Active HDL Simulator or your third party VHDL simulator.
2. Export VHDL file to SpDE.

© 2008 QuickLogic Corporation

3.2.4.3 Using Precision RTL - SpDE

1. Synthesize the VHDL design in Precision RTL.
2. When the Precision RTL compilation is complete, a netlist is created and automatically loaded into SpDE.
3. In SpDE, select your simulator (if you are using a VHDL simulator), or Vital 3.0 Compliant VHDL (if you are using the Active HDL Simulator accessible from QuickWorks).
4. Run tools to optimize, place and route and back annotate.

3.2.4.4 Performing a Postlayout Simulation

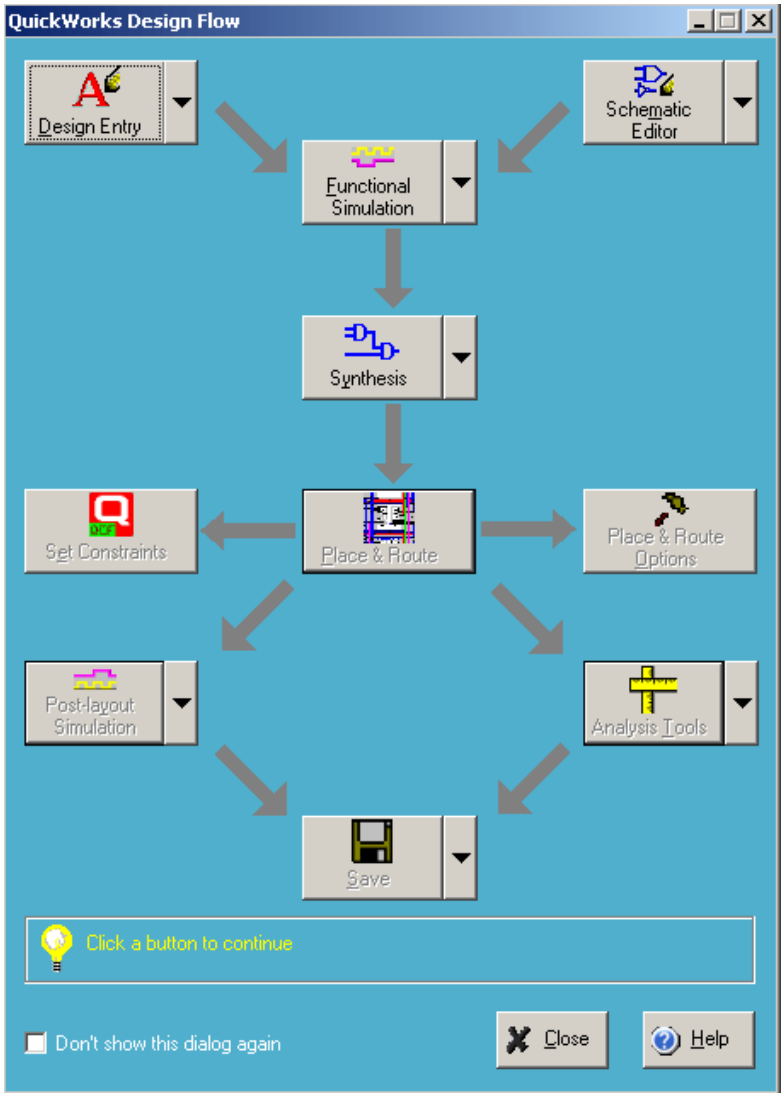
Use your VHDL simulator for postlayout timing simulation.

© 2008 QuickLogic Corporation

3.2.5 QuickWorks Design Flow GUI

The QuickWorks Design Flow interface is intended for novice users. This dialog displays automatically when SpDE is launched (see **Figure 3-5**). The designer can access the main operations of QuickWorks from this dialog without using SpDE menus and toolbar buttons. The QuickWorks Design Flow buttons and arrows give a flow chart appearance and illustrate the design flow. As the designer executes operations, the arrows are highlighted showing the progress. At any time the designer can close this dialog and shift to the normal SpDE menu operations.

Figure 3-5: QuickWorks Design Flow



© 2008 QuickLogic Corporation

3.2.5.1 Disabling/Enabling QuickWorks Design Flow

The QuickWorks Design Flow dialog box displays automatically when SpDE is launched. The dialog box can be closed and opened using the SpDE toolbar button.

The QuickWorks Design Flow status continues to be updated when the display is closed.

To disable this feature:

1. Select the **Do not show this dialog again** checkbox.
2. Click **Close**.

To enable this feature:

1. From the SpDE menu bar, select **View>Preferences.**
2. Select the Windows tab.
3. Select the **Show QW Design Flow at Startup** checkbox.
4. Click **OK**.

3.2.5.2 Using QuickWorks Design Flow

The following briefly describes using the QuickWorks Design Flow feature:

1. When the dialog is displayed for the first time all the buttons, except **Design Entry**, **Schematic Editor**, **Functional Simulation** and **Synthesis**, are disabled.
   - Active HDL Simulator is selected for pre-layout and post-layout simulations
   - Default Text Editor is selected for Design Entry
   - Save Chip File is selected for Save
   - Static Timing Analyzer is selected for Analysis Tools
2. You have an option to start with HDL/Schematic editing, Functional Simulation or Synthesis.
3. To edit the design click on the **Design Entry** or **Schematic Editor** button. To add a new editor or to change the editor click on the pull-down menu button and select the required option.
4. To perform pre-layout/functional simulation click on **Functional Simulation**. The Active HDL Simulator is invoked. To add a new simulator or run any other simulator click on the pull-down menu button and select the required option and then click on **Functional Simulation**.
5. To perform synthesis (using Precision RTL) click on **Synthesis**. The Open file dialog box is displayed. Select a file of type Verilog, VHDL or Precision RTL project file. The Precision RTL tool is invoked. After finishing with the tool, SpDE automatically opens the QuickWorks Design Flow and the **Set Constraints**, **Place & Route Options** and **Place & Route** buttons are enabled. To add a new synthesis tool or to change the synthesis tool click on the pull-down menu button and select the required option.
6. Now you can change the constraints and/or place and route options. After that or without modifying the constraints and/or place and route options you can run placer, router, etc. using **Place & Route**.
7. After the Tools are run the **Save**, **Analysis Tools**, and **Post-Layout Simulator** buttons are enabled. Click on any of these buttons or open another design and start the above flow.

© 2008 QuickLogic Corporation

3.2.5.3 QuickWorks Design Flow Descriptions

Table 3-1 describes each button of the QuickWorks Design Flow in detail.

Table 3-1: QuickWorks Design Flow Options

Button	Description
Design Entry	Opens the selected editor for editing any HDL. The designer can change the editor by clicking the pull-down menu button and select the desired editor from the menu. The selected menu option is ticked. A tool tip shows the selected editor when the designer places the cursor on the button. The designer can also add new editors to the menu using the menu option **Add New Editor**. The designer must enter the name and path of the executable to add a new editor. The editor details are stored permanently and are shown in future sessions of SpDE.
Schematic Editor	Opens the selected editor for editing a schematic. The designer can change the editor by clicking the pull-down menu button and selecting the desired editor from the menu. The selected menu option is ticked. A tool tip shows the selected editor when the designer places the cursor on the button. The designer can also add new editors to the menu using the menu option **Add New Editor**. The designer must enter the name and path of the executable to add a new editor. The editor details are stored permanently and are shown in the menu in future sessions of SpDE.
Functional Simulation	Opens the selected Functional Simulation tool. The designer can change the simulation tool or add a new simulation tool by clicking the pull-down menu button and select the desired tool from the menu. The selected menu option is ticked. A tool tip shows the selected simulation tool when the designer places the cursor on the button. The Active HDL simulation tool is available in this menu, for both Verilog and VHDL simulation. The designer can also add a new simulator to the menu using the menu option **Add new simulator**. The designer must enter the name and path of the executable to add a new simulator. The simulator details are stored permanently and are shown in future sessions of SpDE.
Synthesis	Opens the Precision RTL tool by default. The designer can pass verilog/VHDL/project file to it. It is equivalent to SpDE menu option **File >Import Using Precision RTL.** The designer can add new synthesis tools to the menu using the menu option **Add New Tool**. The designer must enter the name and path of the executable to add a new tool. The tool details are stored permanently and they are shown in the menu in future sessions of SpDE.
Set Constraints	Opens the Constraint Manager. The designer can set placement and timing constraints. Refer Constraint Manager help for details about constraints.
Place & Route Options	Opens the Tools options dialog. It is equivalent to SpDE menu option **Tools>Options**. The designer can change the options of Tools before running them.
Place & Route	Opens the Run Tools dialog. It is equivalent to SpDE menu option **Tools>Run selected tools**. The designer can choose to run the tools Logic Optimizer, Placer, Router, Delay Modeler, Back Annotation, Sequencer.
Post-Layout Simulation	The functionally of this button is equivalent to the **Functional Simulation** button.
Save	Saves selected type of file (chip or LOF). The designer can change the type of file by clicking the pull-down menu button and select the type of file from the menu. The selected type of file (chip or LOF) is shown in the menu by a tick mark. A tool tip shows the file to be saved when the designer places the cursor on the button.
Analysis Tools	Runs the selected tool. The designer can change the selection by clicking on the pull-down menu button and selecting a tool. The selected tool is shown in the menu by a tick mark. The tools shown in the menu are Power Calculator and Static Timing Analyzer. Static Timing Analyzer is the default selection.
Help	Displays the help for the dialog.
Close	Hides the dialog. The designer can reopen it using the tool bar icon .

NOTE: QuickWorks Design Flow does not cover the entire flow of QuickWorks. It covers the main flow of QuickWorks. Some of the operations not covered are the wizards like the Ram/Rom Wizard.

© 2008 QuickLogic Corporation

Chapter 4
Command Line Interface

This chapter explains the TCL Interface and the command line options of QuickWorks. It contains the following sections:

- **“Command Line Options” on page 26**
- **“TCL Interface of QuickWorks - CLI” on page 31**
- **“Running TCL Commands from SpDE GUI” on page 36**
- **“Tables” on page 44**
- **“File Formats” on page 46**
- **“Sample Scripts” on page 51**

The main Quick Works operations are available as Command Line Interfaces (CLIs). These operations can be performed using the command line options or by using the TCL script:

- To use command line options, the user must run `spdecl.exe` with the desired options.
- To use TCL commands, the user has to write a script in a text file and run `spdecl.exe` with the RUNSCRIPT option. The TCL commands can also be executed using TCL shell that comes with `spdecl.exe`.

By using CLI, many operations can be done in a batch mode. The following are some of the benefits in using CLI:

- The tool can be left unattended
- Scripts can be set up so that multiple designs can be run in sequence. This is very useful for running regression tests and benchmarking.
- Scripts can be set up to run the same design with multiple options. This is extremely useful when running a design with various settings because it enables the user to examine the results of the runs and pick the implementation that best fits the user’s needs.

© 2008 QuickLogic Corporation

4.1 Command Line Options

The following command line options are available in QuickWorks.

- IGNORE_CONSTRAINTS: used to load a design without loading the .qcf file
- LOAD: used to load a design (`.qdf` or `.edif` file)
  - PART, PACK: used to retarget a part and a package
- RUNTOTOOL: used to run a tool and its dependent tools
- RUNALLTOOLS: used to run all tools
- SAVE: used to save a `.chp` file
- BACKANNOTATE_ALL: used to run back annotation for all available simulators
- REPORTFILE: used to generate a report file
- PATHANALYZER: used to run the Path Analyzer
- POWERCALC: used to run a Power Calculator
  - INFILE: used to pass the input file for the Power Calculator
  - OUTFILE: used to pass the output file for the Power Calculator
- RAMROM: used to generate ram rom module files
- TIMINGREPORT: used to generate timing report
- GET_TOOL_OPTIONS: used to get predefined tool option values in a file
- SET_TOOL_OPTIONS: used to set supported tool option values through an input file
- GENHEX: used to generate a hex file
- VERSION: used to display the SpDE version
- HELP: used to display help
- RUNSCRIPT: used to run any TCL script
- TCL: used run any TCL command in a TCL shell

For more information, see also:

- **"TCL Interface of QuickWorks - CLI" on page 31**

4.1.1 LOAD and IGNORE_CONSTRAINTS

The CLI supports both `.qdf` and `.edif` files. The design can be loaded with an option of ignoring constraints. If Ignore constraints option is used none of the constraints will be considered while loading the design.

Usage:

```
Spdecl [-IGNORE_CONSTRAINTS] -LOAD <design_name> [-PART <part name> -PACK
<package name>]
```

<design_name> : QDF or EDIF file name with path.

<part_name> : The name of the part to retarget.

<package_name> : The name of the package to retarget.

© 2008 QuickLogic Corporation

Example: The following command loads a qdf file, without loading its constraints. The design is expected from the current directory.

```
spdecl -ignore_constraints -load alu.qdf
```

4.1.2 RUNTOTOOL and RUNALLTOOLS

Correct tool name should be passed along with the command. The valid tool names are: *verifier, logic optimizer, placer, router, delay modeler, back annotation* and *sequencer*. The names are not case sensitive. If the tool name is in two words, it should be given in double quotes. If it is a single word, quotes could be omitted.

`RunAllTools` command runs all the tools in the design flow. The tools are run in the following order:

- Logic optimizer
- Placer
- Router
- Delay modeler
- Sequencer
- Back annotation

Besides the tool given, RunToTool runs the other tools on which the given tool depends.

Both RunToTool and RunAllTools skip the tools, which are already run.

Usage:

```
Spdecl   -RUNTOTOOL <tool_name>
Spdecl   -RUNALLTOOLS
```

<tool_name> : Should be a valid tool name.

Example: The following command loads the design and runs all tools:

```
spdecl -load alu.qdf -runalltools
```

The following command loads the design and runs all tools till Delay Modeler:

```
spdecl -load alu.qdf –runtotool “delay modeler”
```

4.1.3 SAVE

The chip file can be saved with the same name as the design or with a different name (equivalent to 'Save as' option in GUI), using the same CLI command.

Usage:

```
Spdecl -SAVE [<save_filename>]
```

<save_filename> : The chip file name with path. If the file name is not given the chp file will be saved with the same name as the design name and in the same path where the design resides.

Example: The following command loads the design, runs placer and saves the chip file with the same name as the design in the same directory.

```
spdecl -load c:\designs\alu.qdf -runtotool placer -save
```

© 2008 QuickLogic Corporation

4.1.4 BACKANNOTATE_ALL

It runs back annotation for all available simulators.

Usage: `Spdecl -BACKANNOTATE_ALL`

Example: The following command loads the design and runs back annotation for all simulators.

```
spdecl -load alu.qdf -backannotate_all
```

4.1.5 REPORTFILE

The report file will be generated in the same directory where the design resides. The generated file will be an HTML file. The file name is the chip file name with a suffix `rpt`.

Usage: `Spdecl -REPORTFILE`

Example: The following command loads the design, runs placer and saves the chip file and report file in the current directory. The report file name will be alu_rpt.html.

```
spdecl -load alu.qdf -runtotool placer -save alu.chp -reportfile
```

4.1.6 GET_TOOL_OPTIONS

This option is available for selected tool options only. Refer Tool Options table, in **"Tools Options List" on page 44**, for a list of options. The tool options and their values will be saved in a text file in the *ini* file format. Refer to **"File Formats" on page 46** for the file format.

This option is not available as TCL command.

Usage: `Spdecl -GET_TOOL_OPTIONS <out_filename>`

<out_filename> : The file name with path to which the tool options should be saved

Example: The following command gets supported tool options and their values in a file opt.txt.

```
spdecl -get_tool_options opt.txt
```

4.1.7 SET_TOOL_OPTIONS

This option is available only for the tool options specified in the table, in **"Tools Options List" on page 44**. The tool options along with their values should be kept in a file and the file name should be passed with the command line option. So using command line option, values can be set for multiple tool options in one shot. Refer to **"File Formats" on page 46** for the file format. The changed values are persistent, i.e., next time when SpDE is run the modified values are taken. There is one more option, *SetOptionSess*, available as TCL command, which changes the value only in the current session of SpDE.

Usage: `Spdecl -SET_TOOL_OPTIONS <in_filename>`

<in_filename>: File consisting the tool options and values. Refer to **Figure 4.5** for the file format.

Example: The following command sets the tool options specified in file opt.txt.

```
spdecl -set_tool_options opt.txt
```


© 2008 QuickLogic Corporation

4.1.8 POWERCALC

This feature allows calculating power of a part for the given inputs. This can be used with or without design loaded. The inputs are given through an input file. Refer to **"RunToTool and RunAllTools" on page 32** for the file format. The inputs can only be given for calculating FPGA power. At present, inputs for calculating ASSP Power are not supported. When an ASSP design is loaded the inputs will be taken from the design to calculate the ASSP Power. The generated outputs will be stored in a CSV file.

Usage: `Spdecl -POWERCALC [-INFILE <FileName> -OUTFILE <FileName>]`

-INFILE <FileName>: Name and Path of the input file. The extension of the file should be.pwc. See **"RunToTool and RunAllTools" on page 32** for the format of the input file. It is mandatory to give the input file name when design is not loaded, as the part name is read from the input file.

-OUTFILE <FileName>: Name and Path of the file to which output is written. If it is not given, the file name will be <Design Name>.csv when design is loaded and <PartName>.csv when design not loaded. This file will be saved in the current directory.

Example: The following command is for the case where design is loaded. The output file is not given. The file name will be alu.csv and it will be saved in current directory.

```
spdecl -LOAD "Alu.chp" -POWERCALC -INFILE  "C:\spde\powcalin.txt"
```

4.1.9 RAMROM

It takes an input file and generates ram rom module files (`.v, .vhd, .tf, .tb, .mem, .err`). The user has to create an `.rrf` file in the format given in **"Ram Rom Input File" on page 48**.

Usage: `Spdecl -RAMROM <ram-rom config filename>`

< ram-rom config filename >: Name of the input file with path

Example: The following command generates the ram rom module files for the inputs given in the input file.

```
spdecl -ramrom "c:\spde\ram_inputs.txt"
```

4.1.10 TIMINGREPORT

This option is used to generate extended timing report. The options to generate timing report have to be kept in a file in the format given in **"Timing Report Input File" on page 47**. The generated file will be in HTML format and it will have timing information along with basic report.

Usage: `Spdecl -TIMINGREPORT <in filename>`

<in filename>: Name of the input file with path

Example: The following command loads the design, takes input from alu.trt file and generates timing report.

```
spdecl -load "alu.chp" -timingreport "alu.txt"
```

© 2008 QuickLogic Corporation

4.1.11 PATHANALYZER

This runs path analyzer for default options and generates path list in a text file.The output generated is the same as output generated using the **Save Paths** option of the GUI with default selections. If a path analyzer script is present with the design, the script will be executed instead of generating output with default options.

Usage: `Spdecl -PATHANALYZER`

Example: The following command loads the chip file and runs path analyzer.

```
spdecl -load alu.chp -pathanalyzer
```

4.1.12 GENHEX

The Hex file generator is a utility used to map the user specified memory data file onto the actual memory structure of the target devices. The target devices families include QuickRAM, QuickPCI, Eclipse, Eclipse Plus and Eclipse II. The user has to create an input file in the format given in **"Gen Hex Input File" on page 49**.

Usage: `Spdecl -GENHEX <Config filename>`

<Config filename>: Name of the input file with path

Example: The following command takes inputs from the file and generates hex file.

```
spdecl -genhex "hexinput.dat"
```

4.1.13 TCL

This option creates a TCL shell. The user can execute any TCL command including SpDE specific commands explained in the next section. On invoking the TCL shell, the command prompt will be changed to "%". The `exit` command brings you back to the normal prompt.

Usage: `Spdecl -TCL`

© 2008 QuickLogic Corporation

4.2 TCL Interface of QuickWorks - CLI

The Tool Command Language (TCL) is a high-level scripting that supports variables, arrays, lists, data-flow control loops, and procedures for application development. QuickWorks supports version 8.3.4 of TCL.

NOTE: To know more about TCL, refer to http://tcl.activestate.com.

Usage: The user should know basic commands of TCL. The user writes a script in TCL and saves it in a file. The file name should be given as an input to the RUNSCRIPT option of spdecl. TCL shell provided through QuickWorks (`spdecl -TCL`) can also be used, where the user has to give the commands one after the other. The TCL commands are case sensitive.

The SpDE specific commands are:

- **Load**: used to load a design (.qdf or .edif file) with the option of ignoring the constraints and with options to retarget the part and package
- **RunToTool**: used to run a tool and its dependent tools
- **RunAllTools**: used to run all tolls
- **Save**: used to save .chp files
- **GenerateReport**: used to generate a report file
- **RunBA**: used to run back annotation for all available simulators
- **SetOption**: used to set a value for a tool option
- **SetOptionSess**: used to set a value for a tool option for the current session
- **PathAnalyzer:** used to run the Path Analyzer
- **PowerCalc**: used to run the Power Calculator
- **RamRom**: used to generate ram rom module files
- **TimingReport**: used to generate timing report
- **SetXRPTOption**: used to set options for generating timing report

4.2.1 Load

The CLI supports both QDF and EDIF files. The design can be loaded with an option of ignoring constraints. If Ignore constraints option is used none of the constraints will be considered while loading the design.

Usage: `Load <Design name> [<Ignore Constraints Flag>] [<part name> <package name>]`

<Design name>: QDF file or an EDIF file name with path.

<Ignore Constraints Flag>: Should be "TRUE" or "FALSE". If it is TRUE the constraints file will not be loaded and if it is FALSE the constraints file will be loaded. If this value is not specified it will be taken as FALSE by default.

<part_name> : The name of the part to retarget

<package_name> : The name of the package to retarget

Example 1: The following script loads a qdf file, without loading it's constraints. The design is expected from the current directory.

```
Load alu.qdf TRUE
```

© 2008 QuickLogic Corporation

Example 2: The following script loads an EDIF file with constraints. The design is expected from the given path relative to the current working directory.

```
Load ../TestDesigns/pllb_lf_test.edf
```

4.2.2 RunToTool and RunAllTools

Correct tool name should be passed along with the command. The valid tool names are: *verifier, logic optimizer, placer, router, delay modeler, back annotation* and *sequencer.* The names are not case sensitive. If the tool name is in two words, it should be given in double quotes. If it is a single word, quotes could be omitted.

`RunAllTools` command runs all the tools in the design flow. The tools are run in the following order:

- Logic optimizer
- Placer
- Router
- Delay modeler
- Sequencer
- Back annotation

Usage:

```
RunToTool <tool_name>
```

```
RunAllTools
```

<tool_name> : Should be a valid tool name.

Example: The following script loads the design and runs all tools

```
Load alu.qdf
RunAllTools
```

4.2.3 Save

The chip file can be saved with the same name as the design or with a different name (equivalent to **Save As** option in GUI), using the same TCL command. If the file name is different than the design name, the RPT and QCF files will be created with the new name.

Usage: `Save [<save_filename>]`

<save_filename> : The chip file name with path.

NOTE: If the file name is not given, the file will be saved with the same name as the design name and in the same path where the design resides.

Example: The following script loads the design, runs tools till placer and saves the chip file with the same name as the design in the same directory.

```
Load c:/designs/alu.qdf
RunToTool placer
Save
```


© 2008 QuickLogic Corporation

4.2.4 GenerateReport

The report file will be generated in the same directory where the design resides. The generated file will be an HTML file. The file name is the chip file name with suffix `rpt`.

Usage: `GenerateReport`

Example: The following script loads the design, runs tools till placer and saves the chip and report file in the current directory. The report file name will be `alu_rpt.html`.

```
Load alu.qdf
RunToTool placer
Save alu.chp
GenerateReport
```

4.2.5 RunBA

This command runs back annotation for all available simulators.

Usage: `RunBA`

Example: The following script loads the design and runs back annotation for all simulators.

```
Load alu.qdf
RunBA
```

4.2.6 SetOption and SetOptionSess

These options are available only for the tool options specified in the table, in **"Tools Options List" on page 44**. The tool name, option name and value have to passed as parameters. When SetOption is used, the changed values are persistent, i.e., next time when spde is run the modified values are taken. SetOptionSess changes the value only in the current session of spde.

Usage:

`SetOption <Tool_Name> <Option_Name> <Value>`

`SetOptionSess <Tool_Name> <Option_Name> <Value>`

<Tool_Name>: Valid tool name.

<Option_Name>: Valid tool option name. Refer to **Table 4-2 on page 44** for a list of available tool options.

<Value>: The value to be set.

Example: In the following script the placer seed is changed before running placer. The value 53 is stored permanently and next time when the placer is run this value will be used.

```
Load alu.qdf
SetOption placer seed 53
RunToTool placer
Save alu.chp
GenerateReport
```

© 2008 QuickLogic Corporation

4.2.7 PathAnalyzer

This runs the Path Analyzer for default options and generates path list in a text file.The output generated is the same as the output generated using the **Save Paths** option of the GUI with default selections. If a Path Analyzer script is present with the design, the script will be executed instead of generating output with the default options.

Usage: `PathAnalyzer`

Example: The following script loads the chip file and runs path analyzer.

```
Load alu.chp
PathAnalyzer
```

4.2.8 PowerCalc

This feature allows calculating power of a part for the given inputs. This can be used with or without design loaded. The inputs are given through an input file. Refer to **"Power Calculator Input File" on page 46** for the file format. The inputs can only be given for calculating FPGA power. At present, inputs for calculating ESP Power are not supported. When an ESP design is loaded the inputs will be taken from the design to calculate the ESP Power. The generated outputs will be stored in a CSV file.

Usage: `PowerCalc <InputFileName> <OutputFileName>`

<InputFileName> Name and Path of the input file. See **"Power Calculator Input File" on page 46** for the format of the input file. It is mandatory to give the input file name when design is not loaded, as the part name is read from the input file.

<OutputFileName>: Name and Path of the file to which output is written. If it is not given, the file name will be <Design Name>.csv when design is loaded and <PartName>.csv when design not loaded. This file will be saved in the current directory.

Example: The following script is for the case where design is loaded. The input file is not given. For all values defaults will be taken.

```
Load Alu.chp
PowerCalc "" "C:\spde\powcalc.csv"
```

4.2.9 RamRom

It takes an input file and generates ram rom module files (`.v, .vhd, .tf, .tb, .mem, .err`). The user has to create an `.rrf` file in the format given in **"Ram Rom Input File" on page 48**.

Usage: `RamRom <ram-rom config filename>`

<ram-rom config filename>: Name of the input file with path

Example: The following script generates the ram rom module files for the inputs given the input file.

```
RamRom "c:\spde\ram_inputs.rrf"
```


© 2008 QuickLogic Corporation

4.2.10 TimingReport and SetXRPTOption

This option is used to generate extended timing report. The options to generate timing report have to be kept in a file in the format given in **"Timing Report Input File" on page 47**. This command can also be used without giving the input file. In this case, the user needs to set timing report options using the `SetXRPTOption` command. This set option command is available for limited timing report options (see **"Timing Report Options List" on page 45**). The generated file will be in HTML format and it will have timing information along with a basic report.

Usage: `TimingReport [<in filename>]`

<in filename>: Name of the input file with path. If the file name is not mentioned, the options should be set using SetXRPTOption command.

Example: The following script loads the design, takes input from `alu_trt.txt` file and generates timing report.

```
Load "alu.chp"
TimingReport "alu_trt.txt"
```

Usage: `SetXRPTOption <Option name> <Value>`

<Option name>: The name of the option for which the timing report has to be generated. Valid options are listed in **"Timing Report Options List" on page 45**.

< Value >: The value of the option. It should be set to TRUE or FALSE for all options except "NoOfPaths". If nothing is specified, it is treated as TRUE. For the "NoOfPaths" option, the number of paths has to be specified.

Example: The following script loads the design, sets the options and generates timing report.

```
Load "alu.chp"
SetXRPTOption  Clocks  TRUE
SetXRPTOption  NoOfPaths 50
SetXRPTOption  IOPins    # The value has not given, it will be taken as TRUE
TimingReport
```

© 2008 QuickLogic Corporation

4.3 Running TCL Commands from SpDE GUI

SpDE has a facility to execute TCL script files and commands from the GUI.

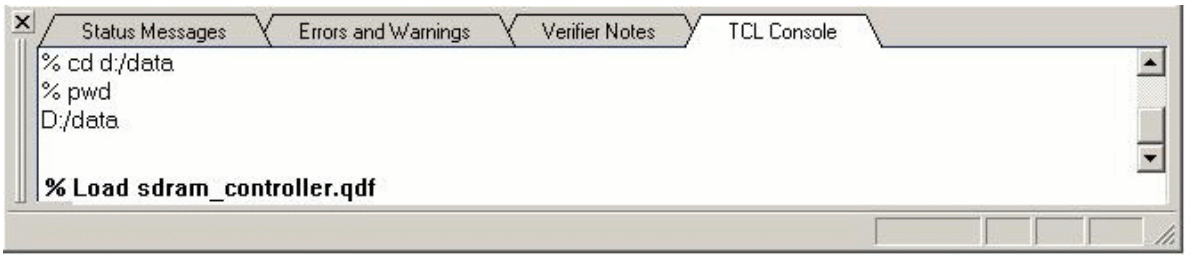
4.3.1 Running TCL Script File

To run the script contained within a file, select **Tools>Run TCL Script**. This is similar to the RUNSCRIPT option in the CLI version of SpDE.

NOTE: All TCL commands are case-sensitive.

4.3.2 Running TCL Command from Command Prompt

TCL commands can be entered at the prompt provided on the TCL Console tab of SpDE's transcript window.



User interactions with SpDE are saved as a list of TCL commands. Command history can be browsed by placing the cursor on the TCL prompt and pressing up or down arrows. The TCL console can automatically complete a partially written SpDE command text.

4.3.3 TCL Commands for GUI

Table 4-1 lists the TCL commands used by the SpDE user interface.

Table 4-1: TCL Commands for SpDE GUI

TCL commands	Description	Usage
Load	Loads a design (`.qdf` or `.edif` file) with options to retarget the part and the package	`Load [<Design name>] [<part name> <package name>]`
RunToTool	Runs a tool and its dependent tools	`RunToTool [<tool_name>]`
RunAllTools	Runs all tools	`RunAllTools`
GenerateReport	Generates a report file	`GenerateReport`
Save	Saves the chip file	`Save [<filename>]`
SaveLOF	Saves the LOF file	`SaveLOF`
RunBA	Runs back annotation for all available simulators	`RunBA`
SetOption	Sets a value for a tool option	`SetOption <Tool_Name> <Option_Name> <Value>`

© 2008 QuickLogic Corporation

Table 4-1: TCL Commands for SpDE GUI (Continued)

TCL commands	Description	Usage
SetOptionSess	Sets a value for a tool option for the current session	`SetOptionSess <Tool_Name> <Option_Name> <Value>`
PowerCalc	Runs the Power Calculator	`PowerCalc [<InputFileName> <OutputFileName>]`
RamRom	Generates a ram-rom module file	`RamRom [<ram-rom config filename>]`
GetToolOptions	Gets the tools options in a given file	`GetToolOptions <out_filename>`
SetToolOptions	Sets the tools options from a given file	`SetToolOptions <in_filename>`
PathAnalyzer	Generates path list	`PathAnalyzer`
StaticTimingAnalyzer	Invokes Static Timing Analyzer tool	`StaticTimingAnalyzer`
ToolsOption	Invokes Tools Option dialog box	`ToolsOption`
ViewPreferences	Invokes View Preference dialog box	`ViewPreferences`
ImportVerilog	Invokes Import Verilog dialog box	`ImportVerilog`
ImportVHDL	Invokes Import VHDL dialog box	`ImportVHDL`
EditSCSini	Invokes Edit SCS ini dialog box	`EditSCSini`
MigrateParts	Invokes Migrate Parts dialog box	`MigrateParts`
UtilizationInfo	Invokes Utilization dialog box	`UtilizationInfo`
ToolsVersions	Invokes Tools Version dialog box	`ToolsVersions`
ReportingPreferences	Invokes Reporting Preference dialog box	`ReportingPreferences`
WebUpdate	Invokes Web Update dialog box	`WebUpdate`
PackageView	Invokes Package View	`PackageView`
ConstraintManager	Invokes Constraint Manager tool	`ConstraintManager`
ESPVersions	Invokes ESP version dialog	`ESPVersions`
PowerSimulator	Invokes Power Simulator dialog	`PowerSimulator`
SetMacroMode	Enables/disables Macro mode	`SetMacroMode [true/false/on/off/0/1]`
SaveAsMacro	Invokes Save As Macro dialog box	`SaveAsMacro`
MacroPlanner	Invokes Macro Planner tool	`MacroPlanner`
Goto	Invokes Goto dialog box	`Goto`
Highlight	Invokes Highlight Net dialog box	`Highlight`
PCIWizard	Invokes the PCI Wizard	`PCIWizard`
EditSchematic	Invokes the Schematic and Hierarchy Navigator tool	`EditSchematic`
EditSymbol	Invokes the Symbol Editor	`EditSymbol`
ActiveHDL	Invokes the Active HDL tool	`ActiveHDL`

© 2008 QuickLogic Corporation

Table 4-1: TCL Commands for SpDE GUI (Continued)

TCL commands	Description	Usage
WaveFormEditor	Invokes the Wave Form Editor	`WaveFormEditor`
UpdateSCS	Invokes the Schematic editor	`UpdateSCS`
Help	Invokes the SpDE help	`Help`
Close	Closes the current design	`Close`
exit or quit	Exits SpDE	`exit or quit`

4.3.3.1 Load

Usage: `Load [<Design name>] [<part name> <package name>]`

<Design name>: QDF file or an EDIF filename with path.

<part_name>: The name of the part to retarget.

<package_name>: The name of the package to retarget.

Example 1: The following script loads a `.qdf` file. The design is expected from the current directory.

```
Load alu.qdf
```

Example 2: The design is expected from the given path relative to the current working directory.

```
Load ../TestDesigns/pllb_lf_test.edf
```

If called without an argument, this command behaves like **File>Open** to open a dialog box for selecting the design to be loaded.

4.3.3.2 RunToTool and RunAllTools

The correct tool name should be passed along with the command. Valid tool names are: *logic optimizer, placer, router, delay modeler, back annotation* and *sequencer*. These names are not case-sensitive. If the tool name consists of two words, it should be enclosed by double quotation marks. If it is a single word, quotation marks are not required.

`RunAllTools` command runs all the tools in the design flow. The tools are run in the following order:

- Logic optimizer
- Placer
- Router
- Delay modeler
- Sequencer
- Back annotation

Usage:

`RunToTool <tool_name>`

`RunAllTools`

<tool_name>: Should be a valid tool name.

© 2008 QuickLogic Corporation

Example: The following script loads the design and runs all tools:

```
Load alu.qdf
RunAllTools
```

If `RunToTool` is called without an argument, the Run Tools dialog box is invoked. This command is equivalent to selecting **Tools>RunSelectedTools...**.

4.3.3.3 Save

Using this command, the chip file can be saved with the same name as the design or with a different name (equivalent to the **Save As** option in GUI). If the file name is different than the design name, the report file and `.qcf` files will be created with the new name.

Usage: `Save [<filename>]`

<filename>: The chip file name with or without path.

NOTE: If the file name is not given, the file will be saved with the same name as the design name and in the same path where the design resides. If the file name is given with a path, the chip file will be saved in the directory of the given path. The report file and `.qcf` files will be created in the directory of the given path.

Example: The following script loads the design, runs tools up to Placer and saves the chip file with the same name as the design in the same directory.

```
Load c:/designs/alu.qdf
RunToTool placer
Save
```

4.3.3.4 GenerateReport

The report file (HTML format) is generated in the same directory where the design resides. The filename is the chip filename with the `.html` extension.

Usage: `GenerateReport`

Example: The following script loads the design, runs tools up to Placer and saves the chip and report files in the current directory. The report filename is `alu_rpt.html`.

```
Load alu.qdf
RunToTool placer
Save
GenerateReport
```

4.3.3.5 SetOption and SetOptionSess

These options are available only for the tool options specified in **Table 4-2**. The tool name, option name and value have to be passed as parameters. When SetOption is used, the changed values are persistent, i.e., next time when SpDE is run, the modified values are taken. SetOptionSess changes the value only in the current session of SpDE.

Usage:

```
SetOption <Tool_Name> <Option_Name> <Value>
SetOptionSess <Tool_Name> <Option_Name> <Value>
```

© 2008 QuickLogic Corporation

<Tool_Name>: Valid tool name.

<Option_Name>: Valid tool option name. Refer to Table 4-1 for a list of available tool options.

<Value>: The value to be set.

Example: In the following script, the Placer seed is changed before Placer is run. The value '53' is stored permanently, and next time when the Placer is run, this value is used.

```
Load alu.qdf
SetOption placer seed 53
RunToTool placer
Save
Report
```

4.3.3.6 PathAnalyzer

This command runs the Path Analyzer with default options and generates the path list in a text file.The generated output is the same as the output generated using the Save Paths option of the GUI with default options. If a Path Analyzer script is present with the design, the script is executed instead of generating output with the default options.

Usage: `PathAnalyzer`

Example: The following script loads the chip file and runs the Path Analyzer:

```
Load alu.chp
PathAnalyzer
```

4.3.3.7 StaticTimingAnalyzer

This command invokes the Static Timing Analyzer tool. It is equivalent to selecting **Tools>Static Timing Analyzer**.

Usage: `StaticTimingAnalyzer`

Example: The following script loads the chip file and invokes the Path Analyzer tool:

```
Load alu.chp
StaticTimingAnalyzer
```

4.3.3.8 PowerCalc

If the command is called without any arguments, the Power Calculator opens. If it is called with an input filename, this feature enables the power calculation of a part based on the provided inputs. This can be used with or without a loaded design. The inputs are given through an input file. Refer to **"Power Calculator Input File" on page 46** for the file format. The inputs can only be given for calculating the FPGA power; inputs for calculating the ESP power are currently not supported. When an ESP design is loaded, the inputs are taken from the design to calculate the ESP power. The generated outputs are stored in a `.csv` file.

Usage: `PowerCalc [<InputFileName> <OutputFileName>]`

<InputFileName>: Name and path of the input file. See **"Power Calculator Input File" on page 46** for the format of the input file. It is mandatory to provide the input filename when a design is not loaded, as the part name is read from the input file.

© 2008 QuickLogic Corporation

<OutputFileName>: Name and path of the file to which output is written. If it is not provided, the file name becomes `<Design Name>.csv` when a design is loaded and `<PartName>.csv` when a design is not loaded. This file is saved in the current directory.

Example: The following script is for the case where design is loaded and the input file is not provided. For all values defaults are taken.

```
Load Alu.chp
PowerCalc "" "C:\spde\powcalc.csv"
```

4.3.3.9 RamRom

This command takes an input file and generates ram-rom module files (`.v`, `.vhd`, `.tf`, `.tb`, `.mem`, `.err`). The user has to create an `.rrf` file in the format given in "Ram Rom Input File" on page 48.

Usage: `RamRom [<ram-rom config filename>]`

<ram-rom config filename>: Name of the input file with path.

Example: The following script generates the ram-rom module files for the inputs given the input file.

```
RamRom "c:\spde\ram_inputs.rrf"
```

NOTE: If input file is not given, the RAM/ROM/FIFO wizard (GUI version) is invoked.

4.3.3.10 TimingReport

This command is used to generate the extended timing report. The options to generate timing report have to be kept in a file, in the format as shown in "Timing Report Input File" on page 47. The generated file is in HTML format and it has timing information along with a basic report.

Usage: `TimingReport [<in filename>]`

<in filename>: Name of the input file with path.

Example: The following script loads the design, takes input from `alu_trt.txt` file and generates timing report.

```
Load "alu.chp"
TimingReport "alu_trt.txt"
```

NOTE: If input file is not given as an argument, the Timing Report dialog box (GUI version) is invoked.

4.3.3.11 GetToolOptions

This option is available for selected tool options only. Refer to Table 4-2 on page 44 for a list of options. The tool options and their values are saved in a text file in the `.ini` format. Refer to "File Formats" on page 46 for more information.

Usage: `GetToolOptions <out_filename>`

<out_filename>: The filename with path to which the tool options should be saved.

© 2008 QuickLogic Corporation

Example: The following command gets supported tool options and their values in a `opt.txt` file.

```
GetToolOptions opt.txt
```

4.3.3.12 SetToolOptions

This option is available only for the tool options specified in **Table 4-2 on page 44**. The tool options along with their values should be kept in a file and the filename should be passed with the command. Values can be set for multiple tool options simultaneously. Refer to **"File Formats" on page 46** for more information. The changed values are persistent, i.e., next time when SpDE is run the modified values are used. There is one more option, `SetOptionSess`, available as TCL command, which changes the value only in the current session of SpDE.

Usage: `SetToolOptions <in_filename>`

<in_filename>: File consisting of tool options and values. Refer to **"File Formats" on page 46** for the file format.

Example: The following command sets the tool options specified in the `opt.txt` file:

```
SetToolOptions opt.txt.
```

4.3.3.13 SetMacroMode

This command enables and disables the Macro Design mode.

Usage: `SetMacroMode [true/false/on/off/0/1]`

Command with arguments `true`, `on`, or `1` enables the Macro Design mode. Arguments other than these disable the mode. The arguments are not case sensitive. If no argument is given, the command enables the mode.

4.3.3.14 Close

Closes the currently open design.

4.3.3.15 exit/quit

Exits the application.

4.3.3.16 Help

Invokes the SpDE help. This command is equivalent to selecting **Help>SpDE**.

4.3.4 Usage of Recording Feature in SpDE GUI

The user interactions with SpDE can be recorded as TCL commands. It is done through the recording feature of SpDE.

© 2008 QuickLogic Corporation

4.3.4.1 Recording Toolbar

4.3.4.2 Start Recording

When this button is pressed, recording mode is enabled and recording of SpDE commands starts. If there are any recorded commands, SpDE allows retaining those commands in the list.

4.3.4.3 Play Recording

This button is enabled when there is no ongoing recording. When this button is pressed, SpDE plays the recorded commands.

4.3.4.4 Stop Recording

This button is enabled when SpDE is in recording mode. After pressing this button, recording is stopped.

If there are any recorded commands, SpDE allows saving the recorded commands in a file, which can be run using the **Tools>Run TCL Script** menu command.

4.3.5 Limitations

Once the play button is pressed to run the recorded TCL commands, the flow of TCL command execution cannot be stopped. Only the commands listed in the **Table 4-1** are supported. All other user interactions with the GUI are not supported.

© 2008 QuickLogic Corporation

4.4 Tables

4.4.1 Tools Options List

Table 4-2: Tools Options List

Tool Name	Option Name	Type	Valid Values
Logic Optimizer	Mode	String	Preliminary, Quality, Overnight
	Level	Integer	0, 1, 2
	Goal	String	Area, Speed
	IgnorePack	Boolean	TRUE, FALSE
Placer	Seed	Integer	-
	Mode	String	Preliminary, Quality, Overnight
	Level	Integer	0, 1, 2
	PowerReduction	Boolean	TRUE, FALSE
*	WindowBasedPlacer	Boolean	TRUE, FALSE
Router	Seed	Integer	-
Delay Modeler	Mode	String	Commercial, Industrial, Military, Custom
	Corner	String	Best, Nominal, Worst
	SpeedGrade	String	Valid Speed Grade
	CustomVCCBest	Float	-
	CustomVCCNominal	Float	-
	CustomVCCWorst	Float	-
	CustomTempBest	Float	-
	CustomTempNominal	Float	-
	CustomTempWorst	Float	-
	CustomVCCLPBest	Float	-
	CustomVCCLPNominal	Float	-
	CustomVCCLPWorst	Float	-
	OutPadCap	Float	-
Back Annotation	FixIOs	Boolean	TRUE, FALSE
	FixFFs	Boolean	TRUE, FALSE
	FixRAMs	Boolean	TRUE, FALSE
	FixECUs	Boolean	TRUE, FALSE
	Simulator	String	Valid Simulator
Verifier	Strip	Boolean	TRUE, FALSE
	RemoveBuffersOnLoad	Boolean	TRUE, FALSE
System	DefaultEDIFPart	String	Valid Part Name
	DefaultEDIFPackage	String	Valid Package Name

© 2008 QuickLogic Corporation

4.4.2 Timing Report Options List

Table 4-3: Timing Report Options List

Option Name	Description
Flip-FlopsClocked	Flip-Flops clocked by non-clock cells
IOPins	IO Pin information
Clocks	Clock signal information
PathsOnClockBoundaries	Paths crossing clock boundaries
NoOfPaths	This is used by the above 3 options.

NOTE:

1. The option names specified above are NOT case sensitive.
2. If *NoOfPaths* is not given, the default value of 50 is used.
3. The options *Clocks* or *PathsOnClockBoundaries*, needs *NoOfPaths* to be set.

© 2008 QuickLogic Corporation

4.5 File Formats

4.5.1 Tool Options File

This section presents the format for the tool options file. This file has sections for each tool with Tool name as the section heading. Each section will have the option names and their values. Below is the format of the file with proper explanation, wherever necessary.

Format:

```
[<Tool Name>]
<Option Name>=<Value>
#Repeat the above line for all options
#Repeat the above block for all Tools
```

Example:

```
[logic optimizer]
Mode=Quality
Level=2
[placer]
Seed=42
Mode=Quality
Level=1
```

4.5.2 Power Calculator Input File

This section presents the format for the input file of Power Calculator. The input file has two parts of data. Main and Clock Information. When any of these values are not given defaults will be taken. The user can skip an entire section. The only value, which is mandatory, is the PartName when the design is not loaded. When a value is given proper validations are done as they are in the GUI. Below is the format of the input file with proper explanation, wherever necessary.

Format:

```
; Input file for Power Calculator
[Main]
; Part Name is mandatory when design is not loaded; Ignored when the design is loaded
   PartName= <PartName>
   VCC=<float value>
   OutputLoading=<integer value>
   VccPll=<float value>
; For PolarPro devices
   NumOfSets=<integer value>
```


© 2008 QuickLogic Corporation

; Clock name will be the section name

```
[<Clock Name>]
SystemFreq=<float value>
```

; Similar sections will be added for other clocks

; The component info

; The clock name and component name will be section name

```
[<Clock Name>-<Component Name>]
AvgFreq=<float value>
Count=<integer value>
BankStd=<DEFAULT|CMOS|LVCMOS|LVCMOS18|
LVCMOS25|GTL|LVTTL|PCI|SSTL2|SSTL3|SSTL18>
VCCIO=<float value>
```

; Valid values for the Component Names are Input, Output, Macro, RAM Block, Clock Buffer, Clock Colbuf, Clock Load, ECU, PLL

Example:

```
[Main]
PartName= "ql4009"
VCC=3.0
OutputLoading=10
[Clock]
SystemFreq=100
[Clock-Output]
AvgFreq=10.0
Count=70
[Clock-RAM Block]
AvgFreq=20.0
Count=7
```

4.5.3 Timing Report Input File

This section explains the format of Timing Report input file. This file contains a single section called *General Info*. At least an option should be given to generate the timing report.

Format:

`[General Info]`

`IOPins=<TRUE or FALSE>` — # IO Pin Information

`Flip-FlopsClocked=<TRUE or FALSE>` — # Flip-Flops clocked by non-clock cells

`Clocks=<TRUE or FALSE>` — # Clock signal information

`PathsOnClockBoundaries=<TRUE or FALSE>` — # Paths crossing clock boundaries

`NoOfPaths=<Number of Paths>` — # To be used by the 2 options above

© 2008 QuickLogic Corporation

Example:

```
[General Info]
IOPins=TRUE
NoOfPaths=50
Clocks= TRUE
```

4.5.4 Ram Rom Input File

This section explains the format of RAM ROM WIZARD CLI input file. Some of the options may not be required, depending on the value given for the option `Module=<RAM/ROM/FIFO>`. Therefore, the requirement of an option depends on the value of the Module option.

Format:

`[RAM ROM FIFO Module]`

`Module=<RAM/ROM/FIFO>`

`Device Family=<Family Name>`

`PartName=QL1A100` # only for PolarPro and further device families

`Ram Size=2` — # only for PolarPro II family

`Depth=<Integer Value>` — # only for RAM and FIFO module for pre-PolarPro families

`Width=<Integer Value>` — # only for RAM and FIFO module for pre-PolarPro families

`PortA_Write_Depth=256` — # only for PolarPro and further device families

`PortA_Write_Width=36` — # only for PolarPro and further device families

`PortB_Read_Depth=256` — # only for PolarPro and further device families

`PortB_Read_Width=36` — # only for PolarPro and further device families

`DualPort=0` — # only for Ram module of PolarPro and further device families

`Count=<Integer Value>` — # only for FIFO module

`Read Mode=<synchronous/Asynchronous>` — # only for RAM/FIFO module

`Speed Optimized=<1/0>` — # only for Sync FIFO module

`AsyncFIFOType=<0/1/2>` — # only for Async FIFO module

`Data Input File=<Input file name>` — # only for ROM module

`RegisterOutput=0` — # only for PolarPro and further device families

`Verilog=<1/0>`

`VHDL=<1/0>`

`Output Directory=<Output Directory Path>`

© 2008 QuickLogic Corporation

Example:

```
[RAM ROM FIFO Module]
Device Family=ArcticLink
PartName=QL1A100
Module=FIFO
PortB_Read_Depth=512
PortB_Read_Width=18
PortA_Write_Depth=512
PortA_Write_Width=18
Read Mode=Asynchronous
RegisterOutput=1
DualPort=0
Verilog=1
VHDL=0
Ouput Directory=e:\test\rrw
```

4.5.5 Gen Hex Input File

This section explains the format for the input file of the command `GenHex`. See the comments on top of each key for a detailed explanation.

Format:

```
[Main]
; Device Type is mandatory, the values can be QUICKRAM, QUICKPCI & QUICKECLIPSE
DeviceType=<device type name>
; No.of Ram Rows available for the device, by default it will be taken as 2
RamRows=<Either 1 or 2>
; Maximum no. of Ram Blocks
MaxCells=<integer value>
; No. of cells to Map
CellToMap=<integer value>
[Mapping]
; This cell name and path values should be specified for all cells to map (i.e., if the user wants to map two cells then there should be two entries for cell name and path)
; Cell Location like A1, B1.
CellName<number>=<alphanumeric value>
; Complete path where the mem file is located.
Path<number>=<path of the mem file>
```

Example:

```
[Main]
DeviceType=QUICKECLIPSE
RamRows=2
MaxCells=20
CellToMap=20

[Mapping]
CellName1=A1
Path1=eclipse.mem
CellName2=B1
Path2=eclipse.mem
CellName3=C1
Path3=eclipse.mem
CellName4=D1
Path4=eclipse.mem
CellName5=E1
Path5=eclipse.mem
CellName6=F1
Path6=eclipse.mem
CellName7=G1
Path7=eclipse.mem
CellName8=H1
Path8=eclipse.mem
CellName9=I1
Path9=eclipse.mem
CellName10=J1
Path10=eclipse.mem
CellName11=A2
Path11=eclipse.mem
CellName12=B2
Path12=eclipse.mem
CellName13=C2
Path13=eclipse.mem
CellName14=D2
Path14=eclipse.mem
CellName15=E2
Path15=eclipse.mem
CellName16=F2
Path16=eclipse.mem
```


© 2008 QuickLogic Corporation

```
CellName17=G2
Path17=eclipse.mem
CellName18=H2
Path18=eclipse.mem
CellName19=I2
Path19=eclipse.mem
CellName20=J2
Path20=eclipse.mem
```

4.6 Sample Scripts

Script 1: The following script loads a design after retargeting, runs all tools till sequencer, runs back annotation for different simulators, saves `sdf` files of different simulators, saves the chip file and runs other analysis tools with proper options. This script covers most of the command line options available in Quick Works.

```
Load feather.qdf FALSE ql6325e ps484
RunToTool "Sequencer"
SetOptionSess "Back Annotation" Simulator "Verilog"
RunToTool "Back Annotation"
eval exec v:/mksnt/mksnt/mv.exe feather.sdf feather_v.sdf
SetOptionSess "Back Annotation" Simulator "Active HDL"
RunToTool "Back Annotation"
eval exec v:/mksnt/mksnt/mv.exe feather.sdf feather_vhd.sdf
PathAnalyzer
Save feather_sq.chp
GenerateReport
## Timing Report Options
SetXRPTOption Clocks TRUE
SetXRPTOption IOPins TRUE
SetXRPTOption PathsOnClockBoundaries TRUE
SetXRPTOption NoOfPaths 50
SetXRPTOption Flip-FlopsClocked TRUE
TimingReport
PowerCalc "" ""
```

© 2008 QuickLogic Corporation

Script 2 (MultiDesign.tcl): The following script defines a procedure which takes a list of designs as arguments, for each design it loads the design, runs tools until delay modeler, saves the chip file, and generate report file. This `proc` can be used in any other `tcl` script after including the file. The sample script shows how to use this `proc`.

```
proc RunMultiDesigns {args} {
foreach { desfile } $args {
if { 0 == [file exists $desfile] } {
puts "Design file $desfile does not exist"
return 0
}
Load $desfile
RunToTool "Delay Modeler"
set rtname [file rootname $desfile]
Save $rtname.chp
GenerateReport
}
}
```

Script 3: This script uses the proc defined above. The TCL command `source` includes other tcl file.

```
source MultiDesign.tcl
RunMultiDesigns  alu.qdf  dual.qdf  feather.qdf  bcu.qdf
```

Script 4 (IterativeRun.tcl): This script runs tools for different placer seeds and saves chip file with different names each time. The seeds are changed for the current session only, so that the default setting will not be affected.

```
set SeedsList [ list 53 59 61 67 71 73 79 83 89 97 101 103\
107 109 113 127 131 137 139 149 151]
set design "e:/testdata/big"
Load "$design.qdf"
set count 0

foreach { seed  } $SeedsList {
  set count [expr $count+1]
  SetOptionSess Placer Seed $seed
  RunToTool "Delay Modeler"
  set chpfile [format "%s_%d.chp" $design $count]
  Save $chpfile
  GenerateReport
  }
```


© 2008 QuickLogic Corporation

Chapter 5
Eclipse Devices

This chapter discusses the following aspects of QuickLogic Eclipse devices:

- **“Overview of Clock Networks” on page 53**
- **“Dedicated Clock Networks” on page 55**
- **“Global Clock Networks” on page 56**
- **“High Drive Networks” on page 58**
- **“Supported I/O Standards in Eclipse Devices” on page 59**
- **“Embedded Computational Units (ECUs)” on page 61**
- **“Configuration Editor” on page 69**
- **“New Pads for Eclipse Devices” on page 70**
- **“Phase Lock Loop (PLL) Macros” on page 73**

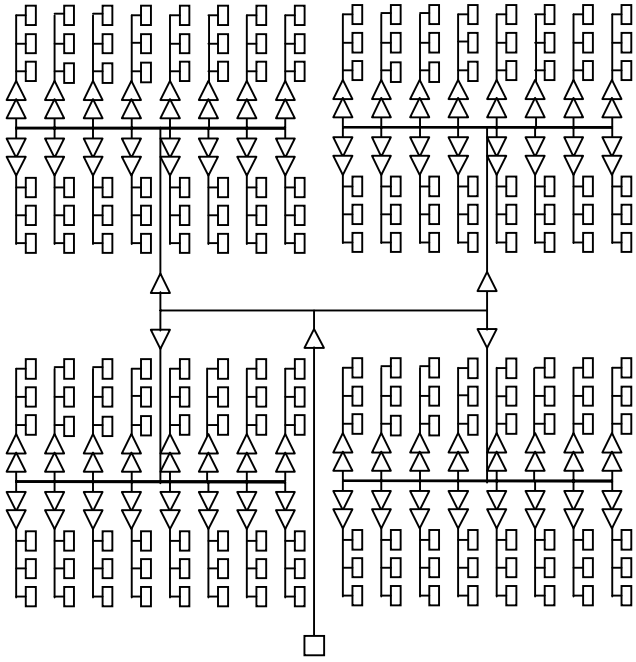
5.1 Overview of Clock Networks

Each QuickLogic Eclipse device contains nine clock networks, one dedicated clock network, and eight global clock networks. Five of the global clock networks can be broken up into a total of twenty quad-net networks. These clock networks are different from the array clock networks and global clock networks in previous QuickLogic devices (QuickPCI, QuickRAM, etc.). The Eclipse clock networks provide more flexibility and are thus more complicated.

© 2008 QuickLogic Corporation

Unlike the clock networks in the previous QuickLogic devices, the Eclipse clock networks are all in the shape of two-level "H" tree (see **Figure 5-1**). Since each device is divided into four quadrants, the first level of each clock tree spans from a clock pad on the edge of the chip to the center of the chip and then to the center of each quadrant. The second level then goes from the center of the quadrant to everywhere inside the quadrant.

Figure 5-1: Eclipse/EclipsePlus Clock Networks



The main advantage of this type of clock network is that it can be divided into four standalone parts (one in each quadrant) and offers the possibility to drive different parts with different signals. Therefore, if one signal on a clock network does not drive logic in the entire chip, e.g. only in two quadrants, the part of this clock network in the other two quadrants can be used for other signals instead of just being wasted.

The dedicated clock network and three of the eight global clock networks must be driven by pins, either directly or through a PLL. The rest five global clock networks can be driven either by pins, or the part of each of these five clock networks inside each quadrant (called a quad-net, quadrant network) can be driven by a separate internal signal.

© 2008 QuickLogic Corporation

5.2 Dedicated Clock Networks

There is one dedicated clock network in each Eclipse family device. This clock is driven by a designated pin and hard-wired to connect to the clock inputs of all logic cells (Super_Cell), all I/O cell registers, and all RAM blocks (both read and write). At every connection, the clock is multiplexed with the programmable clock input, so user can choose whether to use the dedicated clock network to drive the logic or not. This structure has made the load on the dedicated clock network fixed (not programmable) and very balanced, so the dedicated clock network provides fast speed clock distribution with very low skew. See **Table 5-1 on page 55** for the performance numbers of the dedicated clock.

Table 5-1: Dedicated Clock Performance

	Clock Performance	
TT, 25C, 2.5V	Global	Dedicated
Macro (near)	1.51 ns	1.59 ns
I/O (far)	2.06 ns	1.73 ns
Skew	0.55 ns	0.14 ns

The dedicated clock has access to one of the four PLLs. The clock input of this PLL comes from the dedicated clock pin, and its output goes through a similar H-tree to appear at the center of each quadrant. Then, the original input clock (from the pin) and the PLL clock output are multiplexed to drive the clock network inside each quadrant. This multiplexing is done inside each quadrant, so it is possible that the PLL clock output drives the part of the dedicated clock network in some quadrants and the original input clock drives the rest of the quadrants.

The dedicated clock is connected to a designated clock pin on the device. Please consult the device data sheet to verify which pin is the dedicated clock pin for the device and package. If no fix-placement is done for that pin, SpDE automatically assigns the first eligible clock it encounters in the design to the dedicated clock network when it loads a design. If a clock signal is used to drive any non-clock input (e.g. combinatorial logic) in the design, SpDE will not put it onto the dedicated clock network. Instead, it will use a global clock network to drive it.

© 2008 QuickLogic Corporation

5.3 Global Clock Networks

There are eight global clock networks in each Eclipse family device. Each global network allows the following types connections:

- Logic Cell: Clock, Set, Reset, F1, and A2
- RAM Block: Read Clock, Write Clock, Read Enable, Write Enable
- I/O Cell: Clock, Reset, Enable
- ECU (EclipsePlus only): Clock, Reset

Since the global clock networks have programmable connections, the load on each clock network is different in each design. So, the skew of the global clock networks is bigger than the dedicated clock network. Please refer to **Table 5-1 on page 55** for global clock network performance.

Each global clock network can be driven by a pin. Three out of eight global clock networks have access to a separate PLL each. Similar to the dedicated clock network, for each of these three global clock networks the corresponding PLL clock output and the original input clock are multiplexed at each quadrant to drive the clock network inside the quadrant. So it is possible for the part of each global clock network inside some quadrants to be driven by one and the rest driven by the other.

The other five global clock networks are little different. The part of each of these global clock networks inside a quadrant is called a quad-net, so there are 20 quad-nets in total. Each quad-net can either be driven by a pin (the designated pin for the global clock network which the quad-net belongs to), or by any internal signal. This way the five global clock networks have become twenty clock networks, which provide designers with a great amount of flexibility for clock signals and high-fanout signals. However, for any clock signal driving logic in two or more quadrants, it is recommended that the signal comes in through a pin since there tends to be bigger skew if an internal signal is driving two or more quad-nets.

In user designs the *CKPAD_25 um* symbol/macro places an external signal onto a clock network. **Figure 5-2 on page 56** shows the schematic symbol, which should be used in a schematic or mixed schematic/HDL design. This schematic macro is available in the PAD macro library. If you are doing an HDL design you will need to instantiate *CKPAD_25 um* in your design file. The component (VHDL) or module (Verilog) name and port definitions are available in QuickLogic soft macro HDL library (macro.v and macri.vhd).

Figure 5-2: ckpad_25um Schematic Symbol

The *GCLKBUFF_25 um* symbol/macro is used to place any internally generated signal onto a quad-net. SpDE tries to pack all logic that is driven by the same *GCLKBUFF_25 um* into a single quadrant as long as both logic and I/O fit. Otherwise, it needs to drive two or more quad-nets.

Figure 5-3: gclkbuff_25um

© 2008 QuickLogic Corporation

If you are doing an HDL design you will need to instantiate *GCLKBUFF_25um* in your design file. The component (VHDL) or module (Verilog) name and port definitions are available in QuickLogic soft macro HDL library.

© 2008 QuickLogic Corporation

5.4 High Drive Networks

There are 16 high drive networks in each Eclipse family device. They are distributed over eight I/O banks and each bank has two IOCTL pins that can be used to drive two high drive networks. High drive networks can also be driven by internal signals. The allowed connections for high drive networks include clock, reset, and enable inputs of the I/O cells in the same bank, and normal routing. It has quick access to the I/O cells in the same bank, and for driving everything else it uses normal routing where it does not have advantage.

Using an IOCTL pin to drive a high drive network requires using the *HDPAD_25 um* macro (as shown in **Figure 5-4 on page 58**).

Figure 5-4: hdpad_25um

If you need to put an internally generated signal onto a high-drive network, *IO_BUFF_25 um* should be used (as shown in the **Figure 5-5 on page 58**) in the schematic or instantiated in an HDL design.

Figure 5-5: io_buff_25 um

If you use an *IO_BUFF_25 um* resource on a particular IOCTL cell, you will not be able to use an *HDPAD_25 um* on the same IOCTL cell as the resource has already been used. The *IO_BUFF_25 um* schematic symbol can be found in the PAD macro library.

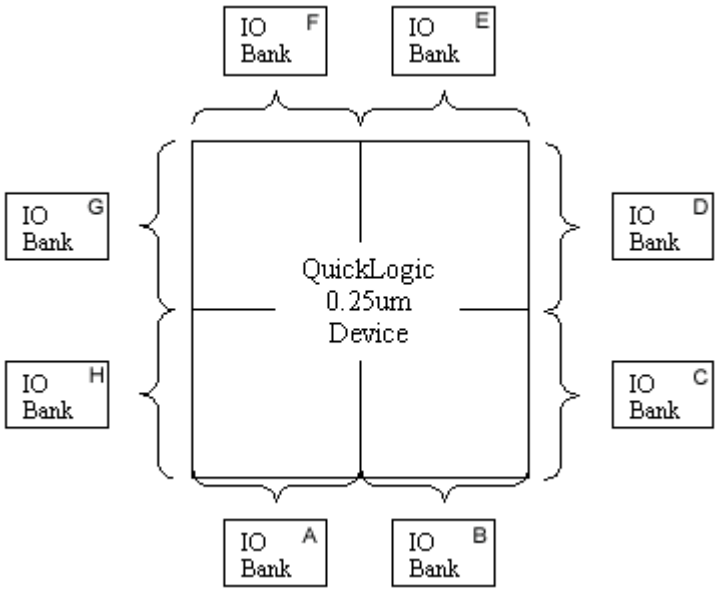
© 2008 QuickLogic Corporation

5.5 Supported I/O Standards in Eclipse Devices

The QuickLogic's Eclipse family of 0.25 µm devices supports a wide variety of I/O standards.

The I/Os in the Eclipse devices are divided into eight different banks (see **Figure 5-6**). Each of these banks can be independently configured with a different standard. This means that each device can interface to several I/O standards at the same time.

Figure 5-6: Eight Different I/O Banks



In order to use these standards the user needs to configure the different I/O banks. The configuration of the I/O banks can be done through the Configuration Editor. The Configuration Editor is an integrated tool in QuickWorks.

Table 5-2: I/O Standard Specification

I/O Standards	Reference voltage	Input Swing	Output Voltage	VCCIO
LVTTL	n/a	0 - 3.3 V	3.3 V	3.3 V
LVCMOS2	n/a	0 - 2. 5V	2.5 V	2.5 V
PCI	n/a	0 - 3.3 V	3.3 V	3.3 V
*GTL+	1.0 V	0.8V - 1.2 V	n/a	2.5 V
*SSTL3	1.5 V	1.30V - 1.70 V	3.3 V	3.3 V
*SSTL2	1.25 V	1.07V - 1.43 V	2.5 V	2.5 V

NOTE: The Eclipse devices support differential input for GTL+, SSTL2 and SSTL3. For differential output support please refer to the application notes at the QuickLogic website. The output characteristic of these I/Os are basically identical to LVTTL, LVCMOS2, and PCI.
Many of these I/O standards are compatible with each other as input only and others as output only. Therefore, the user can have I/Os from different standards within the same bank as long as these standards are compatible with each other.

© 2008 QuickLogic Corporation

5.5.1 Input Standards

For any given I/O banks, the I/O bank properties are defined by whether they are differential input or regular inputs. Therefore, the bank combinations shown in **Table 5-3 on page 60** are allowed:

Table 5-3: Input standard

I/O Standards	Can be Shared with	Why?
LVTTL	PCI	Identical VCCIO (3.3V) and Input Structure
PCI	LVTTL	Identical VCCIO (3.3V) and Input Structure
LVCMOS2	Independent bank	Different VCCIO (2.5V)
GTL+	Independent bank	Input reference voltage is 1.0V (VCCIO = 2.5V)
SSTL3	Independent bank	Input reference voltage is 1.5V (VCCIO = 3.3.V)
SSTL2	Independent bank	Input reference voltage is 1.25V (VCCIO = 2.5V)

For the differential input (GTL+, SSTL2, and SSTL3), the main reason why they need to be independent bank is due to the fact that the reference voltage is common to all I/O's within any given bank. Therefore, even though the VCCIO are identical in some cases, the Eclipse devices will not allow any mixture of GTL+, SSTL2, and SSTL3 input within the same bank. Such as GTL+ and SSTL2 cannot be configured to use the same bank.

5.5.2 Output Standards

Independent of the input standard, the output combinations presented in **Table 5-4 on page 60** are allowed within the same bank:

Table 5-4: Output standard

Type of Outputs	Output Voltage	VCCIO
LVTTL, PCI, SSTL3	3.3V	3.3V
LVCMOS2, GTL+, SSTL2	2.5V	2.5V

© 2008 QuickLogic Corporation

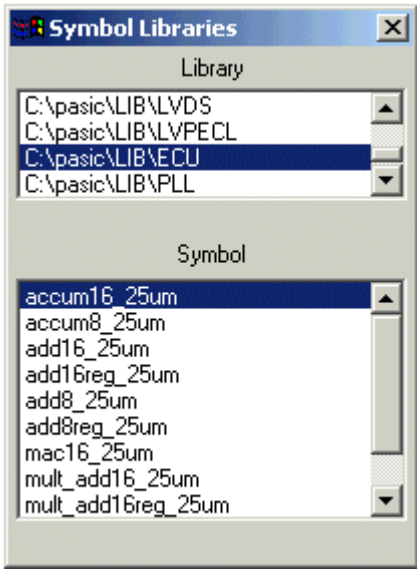
5.6 Embedded Computational Units (ECUs)

This section discusses the following aspects of utilizing ECU modules:

- Incorporating ECU Modules into Schematic Designs
- Incorporating ECU Modules into HDL-Based Designs

5.6.1 Incorporating ECU Modules into Schematic Design

1. Open the **Schematic Editor** in **SpDE**. Refer to the *Mixed Schematic/VHDL Design* chapter of the *QuickWorks Tutorials User Manual* for details on how to use the **Schematic Editor**. For information on how to open the **Symbol Libraries** dialog box to add or incorporate macros and logic cells into schematic designs refer to the *Add Symbols* section of the *Mixed Schematic/VHDL Design* chapter.
2. Use the scroll bar in the **Symbol Libraries** list box, and select **C:\pasic\LIB\ECU** (C:\pasic is the directory where QuickWorks is installed) library for the list of ECU macros.



NOTE: For information on functional descriptions and truth tables for the ECU macros refer to section **"Embedded Computational Units (ECUs)" on page 474**.

The following is a list of the ECU macros available in QuickWorks Schematics Editor: *accum16_25um, accum8_25um, add16_25um, add16reg_25um, add8_25um, add8reg_25um, mac16_25um, mult_add16_25um, mult_add16reg_25um, mult8x8_25um, and mult8x8reg_25um*.

5.6.2 Incorporating ECU Modules into HDL-Based Designs

Refer to the *Verilog-Only Design Tutorial* chapter of the *QuickWorks Tutorials User Manual* for information on how to enter, simulate, place and route, and program QuickLogic devices for Verilog-based designs.

© 2008 QuickLogic Corporation

Refer to the *VHDL-Only Design Tutorial* chapter of the *QuickWorks Tutorials User Manual* for information on how to enter, simulate, place and route, and program QuickLogic devices for VHDL-based designs.

The HDL behavioral model for the ECU block can be obtained from C:\pasic\spde\data\ecu.vhd (where C:\pasic\ is the directory where QuickWorks is installed) for VHDL or from C:\pasic\spde\data\ecu.v (where C:\pasic\ is the directory where QuickWorks is installed) for Verilog. These files contain the VHDL and Verilog behavioral model definition for the ECU entity/ module.

5.6.2.1 ECU Models for Designs Using VHDL

VHDL Port Declaration of the ECU entity

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity ECU is
port (A, B: in std_logic_vector (15 downto 0);
CIN, CLK, RESET, S1, S2, S3, SIGN1, SIGN2: in std_logic;
Q: out std_logic_vector(16 downto 0) );
end ECU;
```


© 2008 QuickLogic Corporation

The following are the port declaration of the ECU entity for the ECU macros available in QuickWorks:

```
ACCUM8_25UM:
entity ACCUM8_25UM is
      Port (   A : In    STD_LOGIC_VECTOR (7 downto 0);
                 CIN : In    STD_LOGIC;
                 CLK : In    STD_LOGIC;
                 RESET : In    STD_LOGIC;
                 Q : Out   STD_LOGIC_VECTOR (8 downto 0) );
end ACCUM8_25UM;

ACCUM16_25UM:
entity ACCUM16_25UM is
      Port (  A : In    STD_LOGIC_VECTOR (15 downto 0);
                 CIN : In    STD_LOGIC;
                 CLK : In    STD_LOGIC;
                 RESET : In    STD_LOGIC;
                 Q : Out   STD_LOGIC_VECTOR (16 downto 0) );
end ACCUM16_25UM;

ADD8_25UM:
entity ADD8_25UM is
      Port (  A : In    STD_LOGIC_VECTOR (7 downto 0);
                 B : In    STD_LOGIC_VECTOR (7 downto 0);
                 CIN : In    STD_LOGIC;
                 Q : Out   STD_LOGIC_VECTOR (8 downto 0) );
end ADD8_25UM;

ADD8REG_25UM:
entity ADD8REG_25UM is
      Port (    A : In    STD_LOGIC_VECTOR (7 downto 0);
                  B : In    STD_LOGIC_VECTOR (7 downto 0);
                CIN : In    STD_LOGIC;
                CLK : In    STD_LOGIC;
                RESET : In    STD_LOGIC;
                Q : Out   STD_LOGIC_VECTOR (8 downto 0) );
end ADD8REG_25UM;
ADD16_25UM:
entity ADD16_25UM is
      Port (    A : In    STD_LOGIC_VECTOR (15 downto 0);
```

© 2008 QuickLogic Corporation

```
                  B : In    STD_LOGIC_VECTOR (15 downto 0);
                  CIN : In    STD_LOGIC;
                  Q : Out   STD_LOGIC_VECTOR (16 downto 0) );
end ADD16_25UM;

ADD16REG_25UM:
entity ADD16REG_25UM is
      Port (    A : In    STD_LOGIC_VECTOR (15 downto 0);
                  B : In    STD_LOGIC_VECTOR (15 downto 0);
                  CIN : In    STD_LOGIC;
                  CLK : In    STD_LOGIC;
                  RESET : In    STD_LOGIC;
                  Q : Out   STD_LOGIC_VECTOR (16 downto 0) );
end ADD16REG_25UM;

MAC16_25UM:
entity MAC16_25UM is
      Port (    A : In    STD_LOGIC_VECTOR (7 downto 0);
                  B : In    STD_LOGIC_VECTOR (7 downto 0);
                  CIN : In    STD_LOGIC;
                  CLK : In    STD_LOGIC;
                  RESET : In    STD_LOGIC;
                  SIGN1 : In    STD_LOGIC;
                  SIGN2 : In    STD_LOGIC;
                  Q : Out   STD_LOGIC_VECTOR (16 downto 0) );
end MAC16_25UM;

MULT8X8_25UM:
entity MULT8X8_25UM is
      Port (    A : In    STD_LOGIC_VECTOR (7 downto 0);
                  B : In    STD_LOGIC_VECTOR (7 downto 0);
                  SIGN1 : In    STD_LOGIC;
                  SIGN2 : In    STD_LOGIC;
                  Q : Out   STD_LOGIC_VECTOR (15 downto 0) );
end MULT8X8_25UM;
MULT8X8REG_25UM:
entity MULT8X8REG_25UM is
      Port (    A : In    STD_LOGIC_VECTOR (7 downto 0);
                  B : In    STD_LOGIC_VECTOR (7 downto 0);
                  CLK : In    STD_LOGIC;
```


© 2008 QuickLogic Corporation

```
                    RESET : In    STD_LOGIC;
                    SIGN1 : In    STD_LOGIC;
                    SIGN2 : In    STD_LOGIC;
                    Q : Out   STD_LOGIC_VECTOR (15 downto 0) );
end MULT8X8REG_25UM;

MULT_ADD16_25UM:
entity MULT_ADD16_25UM is
      Port (    A : In    STD_LOGIC_VECTOR (7 downto 0);
                  B : In    STD_LOGIC_VECTOR (7 downto 0);
                  C : In    STD_LOGIC_VECTOR (15 downto 0);
                  CIN : In    STD_LOGIC;
                  SIGN1 : In    STD_LOGIC;
                  SIGN2 : In    STD_LOGIC;
                  Q : Out   STD_LOGIC_VECTOR (16 downto 0) );
end MULT_ADD16_25UM;

MULT_ADD16REG_25UM:
entity MULT_ADD16REG_25UM is
      Port (       A : In    STD_LOGIC_VECTOR (7 downto 0);
                   B : In    STD_LOGIC_VECTOR (7 downto 0);
                   C : In    STD_LOGIC_VECTOR (15 downto 0);
                 CIN : In    STD_LOGIC;
                 CLK : In    STD_LOGIC;
               RESET : In    STD_LOGIC;
               SIGN1 : In    STD_LOGIC;
               SIGN2 : In    STD_LOGIC;
                   Q : Out   STD_LOGIC_VECTOR (16 downto 0) );
end MULT_ADD16REG_25UM;
```

5.6.2.2 ECU Models for Designs Using Verilog

The following is the Verilog port declaration of the ECU module:

```
module ECU (A, B, CIN, CLK, Q, RESET, S1, S2, S3, SIGN1, SIGN2);
input CLK, RESET, SIGN1, SIGN2, CIN, S1, S2, S3;
input [15:0] A, B;
output [16:0] Q;
```

© 2008 QuickLogic Corporation

The following is the port declaration of the ECU module for the ECU macros available in QuickWorks:

```
accum8_25um:
module accum8_25um( A, CIN, CLK, Q, RESET );

accum16_25um:
module accum16_25um( A, CIN, CLK, Q, RESET );

add8_25um:
module add8_25um( A, B, CIN, Q );

add8reg_25um:
module add8reg_25um( A, B, CIN, CLK, Q, RESET );

add16_25um:
module add16_25um( A, B, CIN, Q );

add16reg_25um:
module add16reg_25um( A, B, CIN, CLK, Q, RESET );

mac16_25um:
module mac16_25um( A, B, CIN, CLK, Q, RESET, SIGN1, SIGN2 );

mult8x8_25um:
module mult8x8_25um( A, B, Q, SIGN1, SIGN2 );

mult8x8reg_25um:
module mult8x8reg_25um( A, B, CLK, Q, RESET, SIGN1, SIGN2 );

mult_add16_25um:
module mult_add16_25um( A, B, C, CIN, Q, SIGN1, SIGN2 );

mult_add16reg_25um:
module mult_add16reg_25um( A, B, C, CIN, CLK, Q, RESET, SIGN1, SIGN2 );
```

The following procedures provide techniques on how to instantiate special QuickLogic ECU buffers into either Verilog or VHDL designs:

For designs using Verilog:

1. Add the file C:\pasic\spde\data\ecu.v (where C:\pasic\ is the directory where QuickWorks is installed) to the Precision TRL Synthesis tool before compilation.

© 2008 QuickLogic Corporation

2. To instantiate the ECU, the Verilog syntax to instantiate the ECU is:

```
ECU instance_name ( .A(input_signals), .B(input_signals),
.CIN(carry_in_signal), .CLK(clock_signal), .Q(output_signals),
.RESET(reset_signal), .S1(S1_signal), .S2(S2_signal), .S3(S3_signal),
.SIGN1(sign1_input_signal), .SIGN2(sign2_input_signal) );
```

For designs using VHDL:

1. Add the file: C:\pasic\spde\data\ecu.vhd (where C:\pasic\ is the directory where QuickWorks is installed) to Precision RTL synthesis tool before compilation.
2. In the Synthesis dialog box, click Add to add a new library named WORK. Make sure to save changes to the project before running synthesis.
3. In the line before the entity declaration to use the ECU, tell the VHDL compiler to use the ql_esp_package package with the following syntax:

```
use work.ql_esp_package.all;
```

4. Inside the architecture, instantiate the ECU with standard VHDL instantiation. The VHDL syntax to instantiate the ECU is:

```
instance_name: ECU Port Map ( A(15 downto 0)=>Input_signals,

B(15 downto 0)=>Input_signals, CIN=>Carry_in_signal, CLK=>Clock_in_signal,
RESET=>reset_in_signal, S1=>S1_signal, S2=>S2_signal, S3=>S3_signal,
SIGN1=>Sign1_input_signal, SIGN2=>Sign2_input_signal,

Q(16 downto 0)=>Output_signal (16 downto 0) );
```

Table 5-5 is a truth table that lists the logic combinations of the 3 one-bit inputs (S1, S2, and S3) with corresponding arithmetic operations, such as multiplication (registered, or unregistered), multiply-add (registered, or unregistered), accumulation, addition (registered, or unregistered), and multiply-accumulate (MAC).

Table 5-5: ECU Arithmetic Operations Truth Table Based on (3) One-bit Inputs (S1, S2, and S3)

Instruction Set Sequencer			Arithmetic Operations
S1	S2	S3	
0	0	0	Multiply[a]
0	0	1	Multiply - Add
0	1	0	Accumulate
0	1	1	Add
1	0	0	Multiply (Registered)[a]
1	0	1	Multiply - Add (Registered)
1	1	0	Multiply - Accumulate
1	1	1	Add (Registered)

a. 8-bit x 8-bit multiply and 8-bit x 8-bit multiply with registered outputs are achieved by setting B[15:0] inputs to zero (ground).

© 2008 QuickLogic Corporation

Table 5-6 lists the I/O ports for the ECU macros and modules.

Table 5-6: Input/Output Ports for ECU Macro and Module

Port	Input/Output	Function
A[7:0] A[15:0]	Input	One of the addend inputs for the 8-bit adder. One of the addend inputs of the 16-bit adder. - Or - Multiplicands A[7:0] and A[15:8] of the 8-bit x 8-bit multiplier.
B[7:0] B[15:0]	Input	One of the addend inputs for the 8-bit adder. One of the addend inputs for the 16-bit adder.
Cin	Input	One-bit Carry-in input for the 16-bit adder, 8-bit adder, 8-bit accumulator, and 16-bit accumulator.
Clk	Input	Clock input for output registers and accumulation operations.
Reset	Input	The signal input that reset output registers and accumulation operations.
Sign1	Input	The signal input that set multiplier A[7:0] for 8-bit x 8-bit multiplication to be unsigned binary number (Sign1 = '0') or signed (2's complement) binary number (Sign1 = '1'). (Refer to Table 5-5)
Sign2	Input	The signal input that set multiplicand A[15:8] for 8-bit x 8-bit multiplication to be unsigned binary number (Sign2 = '0') or signed (2's complement) binary number (Sign2 = '1'). (Refer to Table 5-5)
S1, S2, S3	Inputs	Input signals that configure ECU macros or modules to 8 different modes of arithmetic operations. (Refer to Table 5-1)
Q[16:0] Q[8:0]	Output	17-bit Output for the ECU Q[15:0] is the 16-bit output for either the 16-bit adder, 16-bit accumulator, or 8-bit x 8-bit multiplier. Q[16] is the carry-out bit for the 16-bit adder, or 16-bit accumulator. Q[7:0] is the 8-bit output for either the 8-bit adder, or 8-bit accumulator. Q[8] is the carry-out bit for the 8-bit adder, or 8-bit accumulator.

Table 5-7: Sign-Bits (SIGN1 and SIGN2) Table

Sign-bit	Multiplier	Sign-bit	Multiplicand
SIGN1	A[7:0]	SIGN2	A[15:8]
'0'	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)	'0'	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)
'1'	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)	'1'	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)
'0'	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)	'1'	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)
'1'	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)	'0'	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)

© 2008 QuickLogic Corporation

5.7 Configuration Editor

There are eight I/O banks in each of the Eclipse family of devices. Each I/O bank is independent and each I/O bank has its own VCCIO and VREF supplies. A mixture of I/O standards can be used on the device. **Table 5-8** shows different I/O standards supported and respective reference voltages and output voltage for each I/O standard.

Table 5-8: Different I/O standards and respective reference and output voltage

I/O Standard	Reference Voltage	Input Swing	Output Voltage	VCCIO Voltage
LVTTL	n/a	0-3.3	3.3	3.3
LVCMOS2	n/a	0-2.5	2.5	2.5
PCI	n/a	0-3.3	3.3	3.3
GTL+	1	0.8-1.2	n/a	2.5
SSTL3	1.5	1.07-1.43	3.3	3.3
SSTL2	1.25	1.3-1.7	2.5	2.5

Each I/O pin can be configured by either its net name or its instance name. If any of the I/O pins is configured as differential I/O, you need to assign the bank of the same I/O standard. The bank configured to be a non-differential cannot have any differential I/O in the same bank. The bank configured to support differential I/Os can have non-differential I/Os in the same bank if both I/O standards have compatible voltages. If none of the I/O banks are configured, the SpDE selects the default (LVTTL) standard. For more details, refer to **Chapter 21, "Constraint Manager" on page 267**.

For more information, see also:

- **"Configuration Editor" on page 270** - Using Configuration Editor
- **"Using Constraint Manager" on page 268**

© 2008 QuickLogic Corporation

5.8 New Pads for Eclipse Devices

5.8.1 INPADS

INPAD_25UM should be used on input signals on pins labeled I/O in the pinout tables in the device appendices. The I/O macro INPAD_25UM provides a non-inverting input pad with normal drive current. The pads that have the _25UM appended to the macro name will only work with QuickLogic devices based on 0.25 um process geometry.

If you wish to migrate a design that was originally targeted at an older process geometry, you will need to use the *_25UM* pads. Macros with the *_25UM* suffix will not work on QuickPCI, QuickRAM, or pASIC designs.

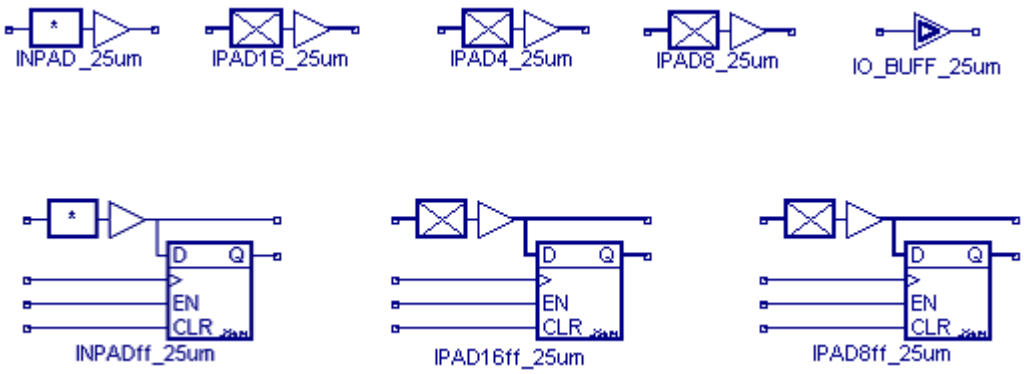
5.8.2 I/O Pads

The QuickLogic Macro Library has a wide selection of I/O pad macros. Included are high-drive pads, clock pads, input pads, output pads, tri-state pads, and bi-directional pads in various configurations.

I/O pads are required in all schematic-based designs. Each QuickLogic device has a different number of pins available. Most pins are standard I/O pins which use the standard INPAD_25um, OUTPAD_25um, TRIPAD_25um, and BIPAD_25um pad macros. Input-only pins require the use of the HDPAD_25um macros. Clock pins require the use of the CKPAD_25um macros.

The macro library also includes bussed versions of the INPAD_25um, OUTPAD_25um, TRIPAD_25um, and BIPAD_25um macros. These macros are named by appending the number of bits to the pad macro name (see **Figure 5-7**). The bussed macros are named IPAD4_25um, IPAD8_25um, IPAD16_25um, IPAD32_25um, etc.

Figure 5-7: I/O Pads



The INPADff and its multiple-instance versions are input pad macros, which are specific to the pASIC3 and QuickRAM devices. They must be used on pins labeled I/O in the pin-out tables shown in the device appendices. The INPADff_25um macro provides the same non-inverting input as the generic INPAD_25um macro, with the addition of the registered version of this signal. In older software versions, the bussed INPADff_25um was named INPADxff.

© 2008 QuickLogic Corporation

5.8.3 HDPAD

The high-drive pads are important resources in the QuickLogic architecture, providing a fast connection to the IO cells. An example of HDPAD is shown in

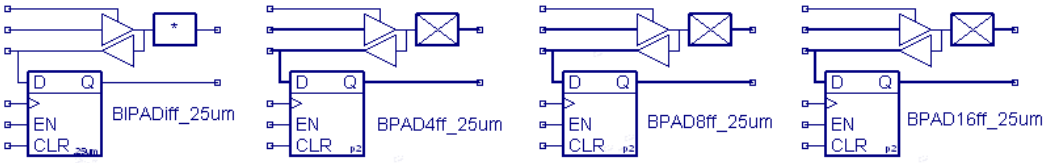
Figure 5-8: HDPAD_25um

5.8.4 BIPADS

BIPADs (see **Figure 5-9**) can only be used on pins labelled I/O in the pinout tables. QuickWorks now includes several additions to the existing library of BIPADs which function in the EclipsePlus and Eclipse architecture. The enable signal on the bipads is active high. When enable is low, the output buffer is tri-stated and the pad is in the input state. When the enable is high, the BIPAD is in the output state. QuickWorks supports BIPADs of several widths, including 4-bit, 8-bit, and 16-bits of data (see **Figure 5-10**).

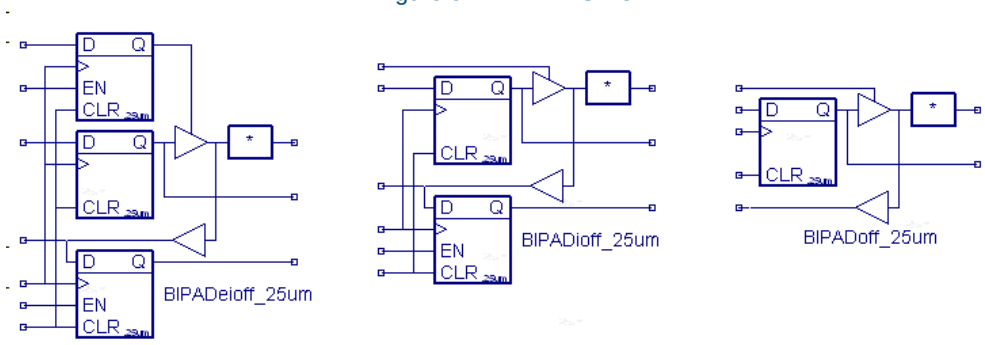
Figure 5-9: BIPADS—1

Figure 5-10: BIPADS—2



The pads shown in **Figure 5-11** (BIPADXff_25um pads) can be used when the inputs from the bipads needs to be registered. These pads include a D flip-flop to provide the registered inputs. These pads are available for 4-bit, 8-bit and 16-bit wide data.

Figure 5-11: BIPADS—3

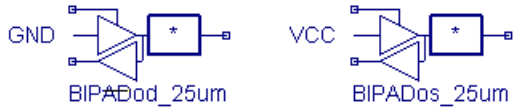


The pads shown in **Figure 5-11** support registering either the input or the output path of the BIPAD. BIPADeioff_25um is used when input, output, and the enable signals to the BIPAD are registered. BIPADioff_25um is used when both the input and the output paths are registered and BIPADoff_25um is used when the output path is alone registered.

© 2008 QuickLogic Corporation

BIPADod_25um (see **Figure 5-12**) acts in an open drain pad, in that the output from the BIPAD is either low or tri-stated. BIPADos_25um is a BIPAD in open source pad, and hence the output is either high or tri-stated.

Figure 5-12: BIPADS—4

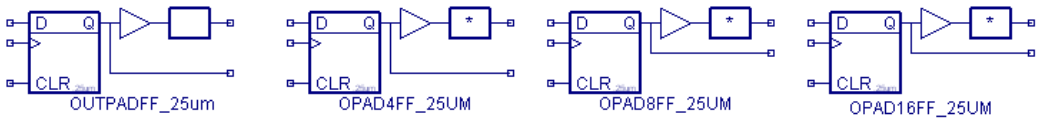


5.8.5 OUTPADs

OUTPADs should be used on pins labeled I/O. The OUTPADs in QuickWorks support various widths of data, including 4-bit, 8-bit and 16-bits of data (see **Figure 5-13** and **Figure 5-14**).

Figure 5-13: OUTPADS—1

Figure 5-14: OUTPADS—2

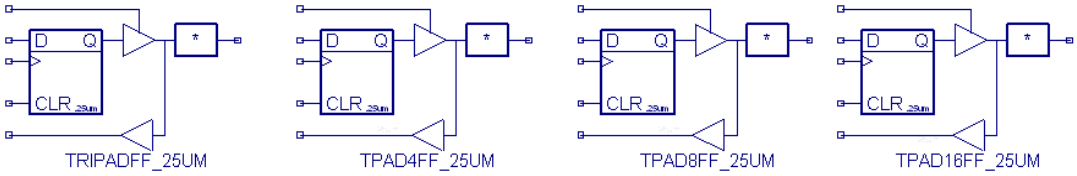


5.8.6 TRIPADs

The tripads (see **Figure 5-15** and **Figure 5-16**) are the list of additions to the existing TRIPADs. The enable on the tripad is active high - when enable is low the pad is in tristate; when the enable is high the pad is in the output state.

Figure 5-15: TRIPADS—1

Figure 5-16: TRIPADS—2



The TRIPADs presented in **Figure 5-16 on page 72** include a flip-flop on the output path of the TRIPAD so that the output can be registered. There are four TRIPAD macros in the schematic macro library, each supporting a different width, TPADFF_25um, TPAD4FF_25um, TPAD8FF_25um and TPAD16FF_25um.

© 2008 QuickLogic Corporation

5.9 Phase Lock Loop (PLL) Macros

All QuickLogic's 0.25 um devices have embedded Phase Locked Loops (PLLs) to enable users to reduce the clock-to-out timing of their design and to run the internal logic of the device at a faster or slower rate than the incoming clock frequency.

Therefore, there is no clock delay between the input clock signal and clock signal coming from the clock network or driving off-chip devices (PLL will compensate for delay in the clock network). Using time domain multiplexing, PLLs enable users to improve device area efficiency by sharing resources within the device.

The PLLS can be used in eight distinct modes of operation. **Table 5-9** indicates these models and offers a brief description of each model.

NOTE: HF represents high frequency and LF represents low frequency.

Table 5-9: PLL Models and Descriptions

PLL MODELS	Output Frequency	Input Frequency Range	Output Frequency Range
PLL_HF	Same as Input	66 MHz - 250 MHz	66 MHz - 250 MHz
PLL_LF	Same as Input	25 MHz - 66 MHz	25 MHz - 66 MHz
PLL_MULT2HF	2x	33 MHz - 125 MHz	66 MHz - 250 MHz
PLL_MULT2LF	2x	12.5 MHz - 33 MHz	25 MHz - 66 MHz
PLL_DIV2HF	1/2x	250 MHz - 500 MHz	125 MHz - 250 MHz
PLL_DIV2LF	1/2x	50 MHz - 250 MHz	25 MHz - 125 MHz
PLL_MULT4	4x	12.5 MHz - 50 MHz	50 MHz - 200 MHz
PLL_DIV4	1/4x	100 MHz - 500 MHz	25 MHz - 125 MHz

Table 5-10 presents a brief description of the PLL signals.

Table 5-10: PLL Signal Descriptions

PLLCLK_IN	Input clock signal
PLL_RESET	If PLL_RESET is 1, then CLKNET_OUT and CLKPAD_OUT are reset to 0 (This signal has to be asserted in order for the LOCK_DETECT to work).
ONn_OFFCHIP	This signal is used to select whether the PLL will drive the internal clock network or used off chip and can only be tied to GND or VCC. This signal needs to be tied to GND to drive internal gates.This signal needs to be tied to VCC for off-chip use.This is NOT a dynamic signal but a static signal.
PLLCLK_OUT	Can drive internal gates when ONn_OFFCHIP is tied to GND
CLKNET_OUT	Signal bypassing PLL and can drive internal gates when ONn_OFFCHIP is tied to GND. However, this signal can not be used in the same quadrant where the PLL signal is used.
PLLPAD_OUT	Signal to be used off-chip and can only be used when ONn_OFFCHIP signal is tied to VCC.
LOCK_DETECT	Signal for lock detection. (NOTE: For simulation, this signal gets asserted after 10 clock cycles. However, it takes a maximum of 200 clock cycles to sync up with input clock upon release of RESET signal).

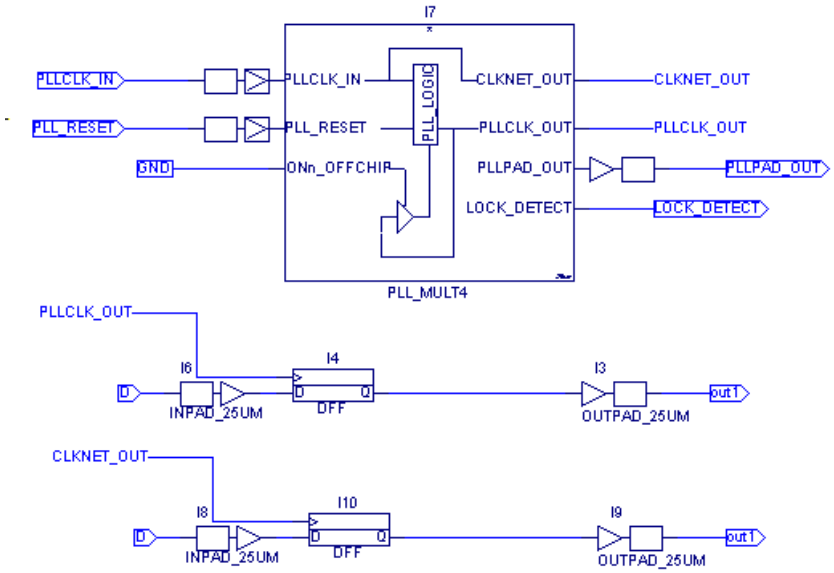
© 2008 QuickLogic Corporation

NOTE: PLLCLK_IN and PLL_RESET signals have INPAD, and CLKPAD_OUT has OUTPAD. (The user does not have to add additional pads.)

5.9.1 Schematic Entry Example

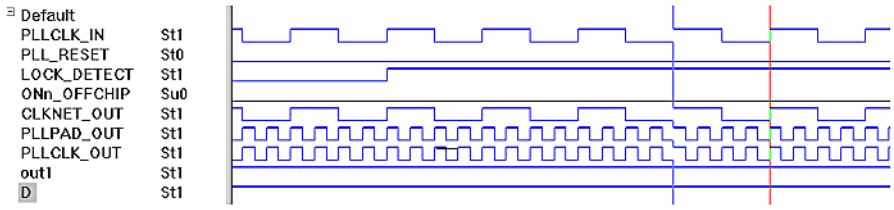
Figure 5-17 is an example of PLL_MULT2HF driving two D-flip-flops.

Figure 5-17: PLL_MULT2HF Example



5.9.2 Prelayout Simulation Waveform

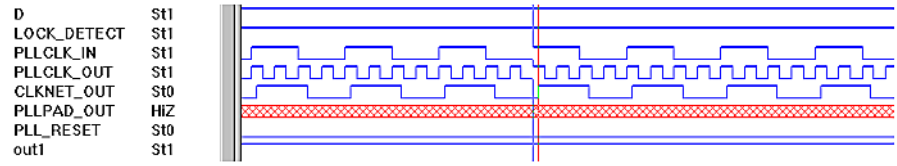
Figure 5-18: Prelayout Simulation Waveform



Input frequency is 50 MHz and all of the output clock edges of PLLs line up.

5.9.3 Postlayout Simulation Waveform

Figure 5-19: Postlayout Simulation Waveform



There is a delay of about 1.1 ns on the CLKNET_OUT signal as it is using the clock tree.

© 2008 QuickLogic Corporation

5.9.4 Using PLL Dedicated Pins

Users must use the dedicated PLL pins when using PLLs:

- VCCPLL needs to be tied to 2.5 V.
- GNDPLL needs to be tied to 0 V.
- PLLIN is used for PLLCLK_IN signal.
- PLLOUT is used for PLLPAD_OUT signal.
- PLLRST is used for PLL_RESET signal.

5.9.5 Disabling PLLs

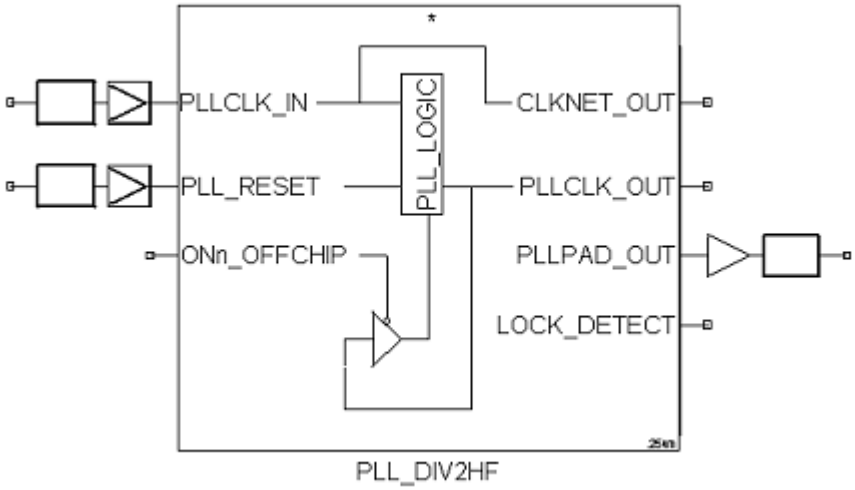
If left unconnected, the unused PLLs can draw a large amount of current. It is recommended that the users connect the unused PLL dedicated pins as presented in the following list to disable the PLLs.

- PLLRST needs to be tied to VCC.
- VCCPLL needs to be tied to 2.5 V.
- GNDPLL needs to be tied to 0 V.

NOTE: Contact QuickLogic regarding the specification on jitters.

5.9.5.1 PLL_DIV2HF

Figure 5-20: PLL_DIV2HF



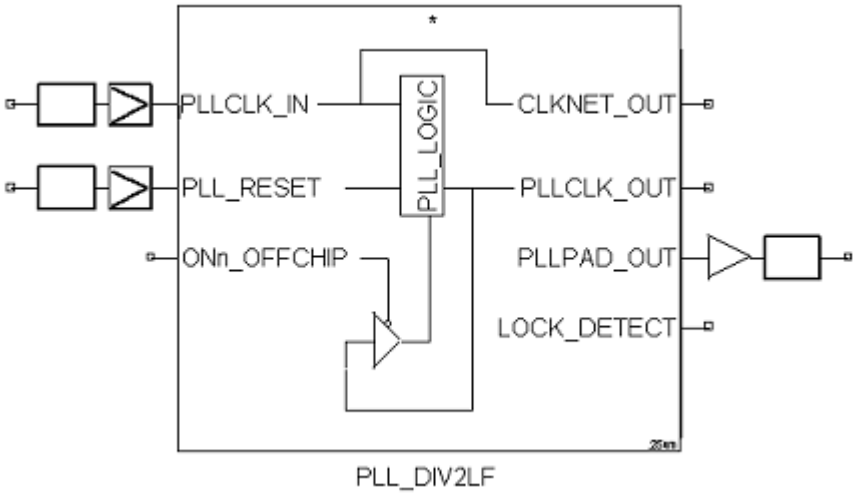
INPUTS: PLLCLK_IN, PLL_RESET, ONN_OFFCHIP
OUTPUTS: PLLCLK_OUT, CLKNET_OUT, PLLPAD_OUT, LOCK_DETECT

Table 5-11: PLL_DIV2HF Inputs and Outputs

PLLCLK_ IN	PLL_ RESET	ONn_ OFFCHIP	CLKNET_ OUT	PLLCLK_ OUT	PLLPAD_ OUT	LOCK_ DETECT
Clock Frequency between 250 MHz to 500 MHz	H	X	PLLCLK_ IN	L	L	0
Clock Frequency between 250 MHz to 500 MHz	L	X	PLLCLK_INClock Frequency between 250 MHz to 500 MHz	PLLCLK_IN/ 2 Clock Frequency between 125 MHz to 250 MHz	PLLCLK_IN/ 2 Clock Frequency between 125 MHz to 250 MHz	1

5.9.5.2 PLL_DIV2LF

Figure 5-21: PLL_DIV2LF



INPUTS: PLLCLK_IN, PLL_RESET, ONN_OFFCHIP
OUTPUTS: PLLCLK_OUT, CLKNET_OUT, PLLPAD_OUT, LOCK_DETECT

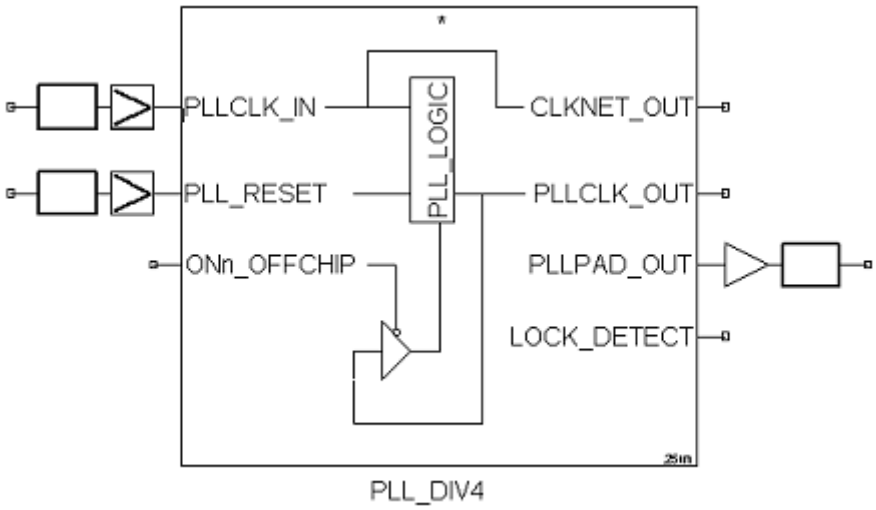
Table 5-12: PLL_DIV2LF Inputs and Outputs

PLLCLK_ IN	PLL_ RESET	ONn_ OFFCHIP	CLKNET_ OUT	PLLCLK_ OUT	PLLPAD_ OUT	LOCK_ DETECT
Clock Frequency between 50 MHz to 250 MHz	H	X	PLLCLK_IN Clock Frequency between 50 MHz to 250 MHz	L	L	0
Clock Frequency between 50 MHz to 250 MHz	L	X	PLLCLK_IN Clock Frequency between 50 MHz to 250 MHz	PLLCLK_IN/ 2 Clock Frequency between 25 MHz to 125 MHz	PLLCLK_IN/ 2 Clock Frequency between 25 MHz to 125 MHz	1

© 2008 QuickLogic Corporation

5.9.5.3 PLL_DIV4

Figure 5-22: PLL_DIV4



INPUTS: PLLCLK_IN, PLL_RESET, ONN_OFFCHIP
OUTPUTS: PLLCLK_OUT, CLKNET_OUT, PLLPAD_OUT, LOCK_DETECT

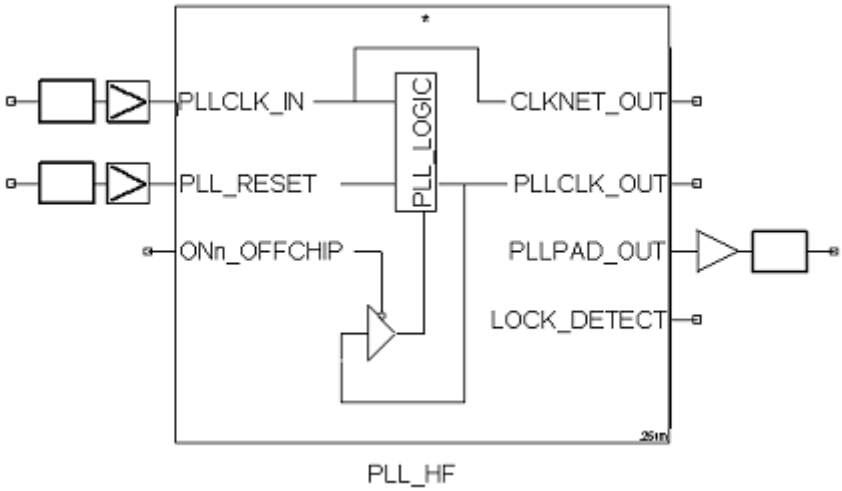
Table 5-13: PLL_DIV4 Inputs and Outputs

PLLCLK_ IN	PLL_ RESET	ONn_ OFFCHIP	CLKNET_ OUT	PLLCLK_ OUT	PLLPAD_ OUT	LOCK_ DETECT
Clock Frequency between 100 MHz to 500 MHz	H	X	PLLCLK_IN Clock Frequency between 100 MHz to 500 MHz	L	L	0
Clock Frequency between 100 MHz to 500 MHz	L	X	PLLCLK_IN Clock Frequency between 100 MHz to 500 MHz	PLLCLK_IN/ 4 Clock Frequency between 25 MHz to 125 MHz	PLLCLK_IN/ 4 Clock Frequency between 25 MHz to 125 MHz	1

© 2008 QuickLogic Corporation

5.9.5.4 PLL_HF

Figure 5-23: PLL_HF



INPUTS: PLLCLK_IN, PLL_RESET, ONN_OFFCHIP
OUTPUTS: PLLCLK_OUT, CLKNET_OUT, PLLPAD_OUT, LOCK_DETECT

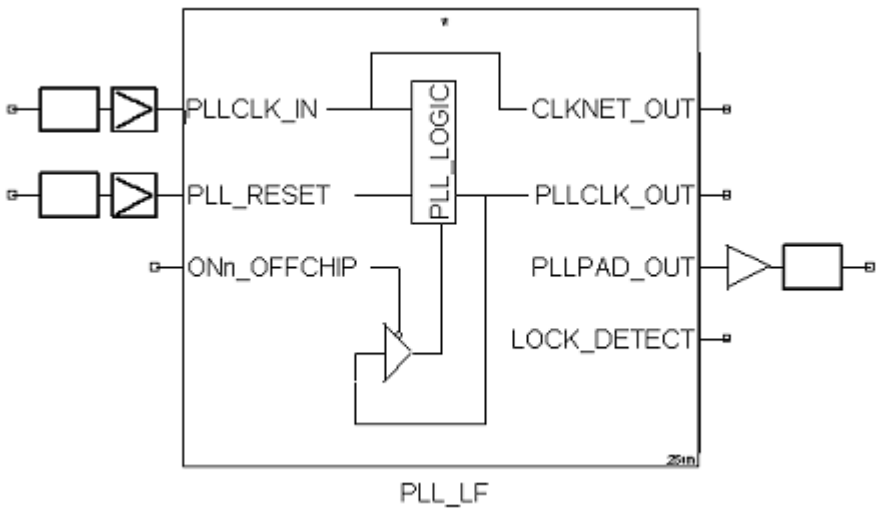
Table 5-14: PLL_HF Inputs and Outputs

PLLCLK_ IN	PLL_ RESET	ONn_ OFFCHIP	CLKNET_ OUT	PLLCLK_ OUT	PLLPAD_ OUT	LOCK_ DETECT
Clock Frequency between 66 MHz to 250 MHz	H	X	PLLCLK_IN Clock Frequency between 66 MHz to 250 MHz	L	L	0
Clock Frequency between 66 MHz to 250 MHz	L	X	PLLCLK_IN Clock Frequency between 66 MHz to 250 MHz	PLLCLK_IN Clock Frequency between 66 MHz to 250 MHz	PLLCLK_IN Clock Frequency between 66 MHz to 250 MHz	1

© 2008 QuickLogic Corporation

5.9.5.5 PLL_LF

Figure 5-24: PLL_LF



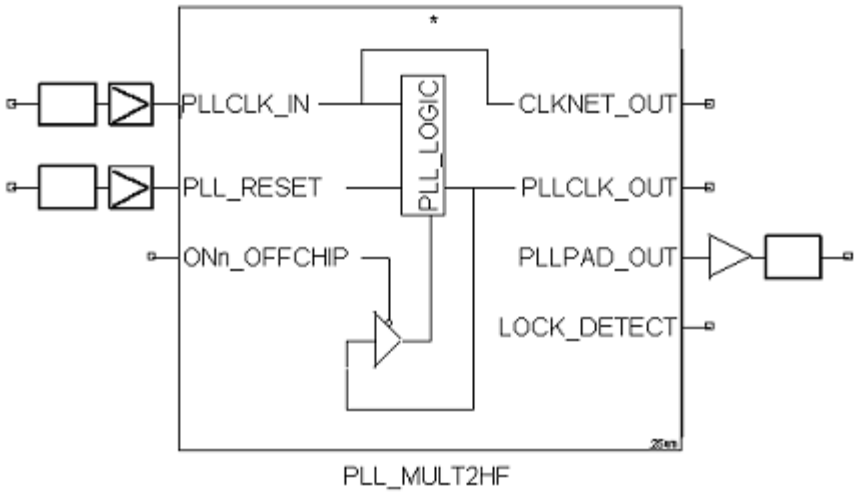
INPUTS: PLLCLK_IN, PLL_RESET, ONN_OFFCHIP
OUTPUTS: PLLCLK_OUT, CLKNET_OUT, PLLPAD_OUT, LOCK_DETECT

Table 5-15: PLL_LF Inputs and Outputs

PLLCLK_ IN	PLL_ RESET	ONn_ OFFCHIP	CLKNET_ OUT	PLLCLK_ OUT	PLLPAD_ OUT	LOCK_ DETECT
Clock Frequency between 25 MHz to 66 MHz	H	X	PLLCLK_IN Clock Frequency between 25 MHz to 66 MHz	L	L	0
Clock Frequency between 25 MHz to 66 MHz	L	X	PLLCLK_IN Clock Frequency between 25 MHz to 66 MHz	PLLCLK_IN Clock Frequency between 25 MHz to 66 MHz	PLLCLK_IN Clock Frequency between 25 MHz to 66 MHz	1

5.9.5.6 PLL_MULT2HF

Figure 5-25: PLL_MULT2HF



INPUTS: PLLCLK_IN, PLL_RESET, ONN_OFFCHIP
OUTPUTS: PLLCLK_OUT, CLKNET_OUT, PLLPAD_OUT, LOCK_DETECT

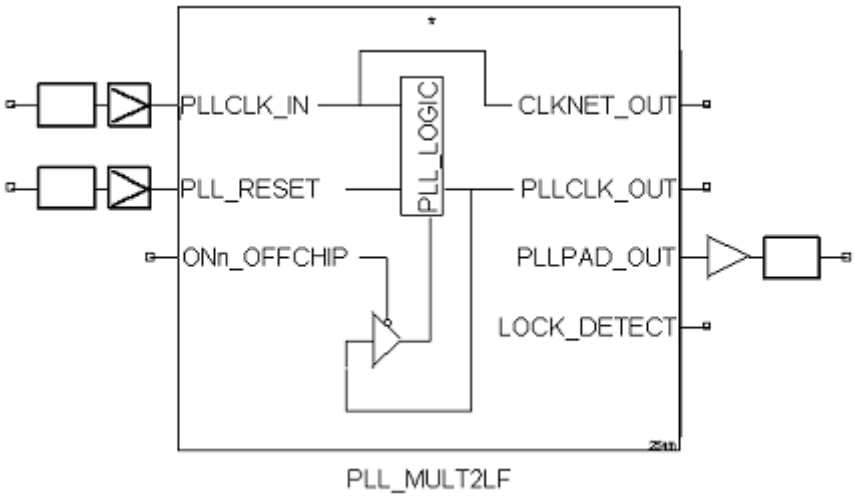
Table 5-16: PLL_MULT2HF Inputs and Outputs

PLLCLK_ IN	PLL_ RESET	ONn_ OFFCHIP	CLKNET_ OUT	PLLCLK_ OUT	PLLPAD_ OUT	LOCK_ DETECT
Clock Frequency between 33 MHz to 125 MHz	H	X	PLLCLK_IN Clock Frequency between 33 MHz to 125 MHz	L	L	0
Clock Frequency between 33 MHz to 125 MHz	L	X	PLLCLK_IN Clock Frequency between 33 MHz to 125 MHz	PLLCLK_IN x 2 Clock Frequency between 66 MHz to 250 MHz	PLLCLK_IN x 2 Clock Frequency between 66 MHz to 250 MHz	1

© 2008 QuickLogic Corporation

5.9.5.7 PLL_MULT2LF

Figure 5-26: PLL_MULT2LF



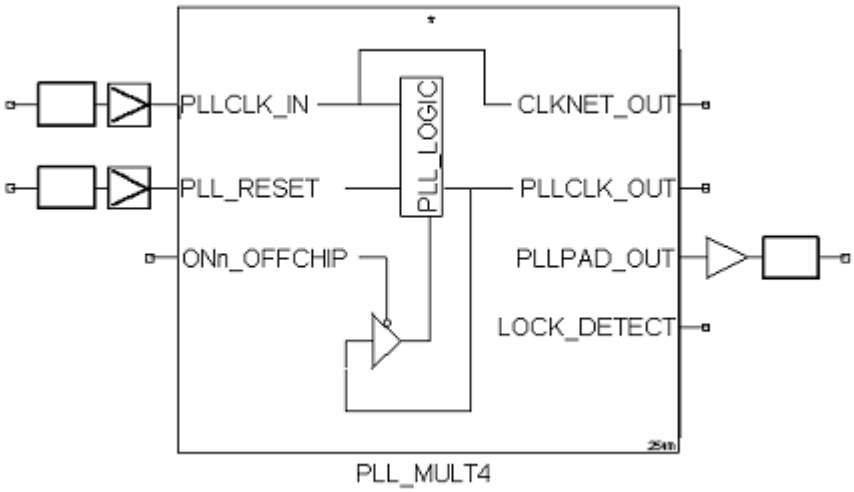
INPUTS: PLLCLK_IN, PLL_RESET, ONN_OFFCHIP
OUTPUTS: PLLCLK_OUT, CLKNET_OUT, PLLPAD_OUT, LOCK_DETECT

Table 5-17: PLL_MULT2LF Inputs and Outputs

PLLCLK_ IN	PLL_ RESET	ONn_ OFFCHIP	CLKNET_ OUT	PLLCLK_ OUT	PLLPAD_ OUT	LOCK_ DETECT
Clock Frequency between 12.5 MHz to 33 MHz	H	X	PLLCLK_IN Clock Frequency between 12.5 MHz to 33 MHz	L	L	0
Clock Frequency between 12.5 MHz to 33 MHz	L	X	PLLCLK_IN Clock Frequency between 12.5 MHz to 33 MHz	PLLCLK_IN x 2 Clock Frequency between 25 MHz to 66 MHz	PLLCLK_IN x 2 Clock Frequency between 25 MHz to 66 MHz	1

5.9.5.8 PLL_MULT4

Figure 5-27: PLL_MULT4



INPUTS: PLLCLK_IN, PLL_RESET, ONN_OFFCHIP
OUTPUTS: PLLCLK_OUT, CLKNET_OUT, PLLPAD_OUT, LOCK_DETECT

Table 5-18: PLL_MULT4 Inputs and Outputs

PLLCLK_IN	PLL_RESET	ONn_OFFCHIP	CLKNET_OUT	PLLCLK_OUT	PLLPAD_OUT	LOCK_DETECT
Clock Frequency between 12.5 MHz to 50 MHz	H	X	PLLCLK_IN Clock Frequency between 12.5 MHz to 50 MHz	L	L	0
Clock Frequency between 12.5 MHz to 50 MHz	L	X	PLLCLK_IN Clock Frequency between 12.5 MHz to 50 MHz	PLLCLK_IN x 4 Clock Frequency between 50 MHz to 200 MHz	PLLCLK_IN x 4 Clock Frequency between 50 MHz to 200 MHz	1

© 2008 QuickLogic Corporation

Chapter 6
PolarPro Devices

This chapter includes the following topics:

- **"PolarPro Overview" on page 83**
- **"Features and Benefits" on page 83**
- **"PolarPro Resources" on page 84**

6.1 PolarPro Overview

The PolarPro FPGA technology was specifically designed to meet the interconnect and system logic requirements of power-sensitive and portable applications. Through a new and innovative logic cell architecture, versatile embedded memory with built-in FIFO control logic and advanced clock management control units, the PolarPro architecture is synthesis friendly and logic mapping efficient, enabling cost-effective and ultra-low power logic designs.

6.2 Features and Benefits

Table 6-1 describes the main features of the PolarPro device family architecture.

Table 6-1: Features and Benefits of the PolarPro Architecture

Features	Benefits
Enhanced Logic Cell Architecture	A flexible, synthesis friendly logic cell that enables efficient mapping of up to 13-bit wide input functions, as well as mapping of any 4-input LUT, or two 3-input LUT combinations into a single level of logic. Each logic cell has four simultaneous outputs and a dedicated Enable D-Type Flip Flop.
VLP Mode	From an external input control pin, the FPGA device can be put into Very Low Power (VLP) mode, in which the device will typically draw less than 10 µA. Within the VLP mode, I/O states and internal register values are retained. This capability provides an instant ability to save battery power when the device function is not needed.
Flexible Clock Networks	There are five clocks in each of the four quadrants. This allows for clocking on one or more quadrants, or on a global clock basis, resulting in a more efficient use of resources.
Programmable I/Os	General purpose programmable I/Os with fully registered Input/Outputs. Each I/O has a programmable Slew Rate control, programmable Pull-up/Pull-down and Weak Keeper functionality. Support for SSTL3, SSTL2, SSTL18, LVTTL, LVCMOS up to 200 MHz and PCI 2.3 up to 66 MHz.

© 2008 QuickLogic Corporation

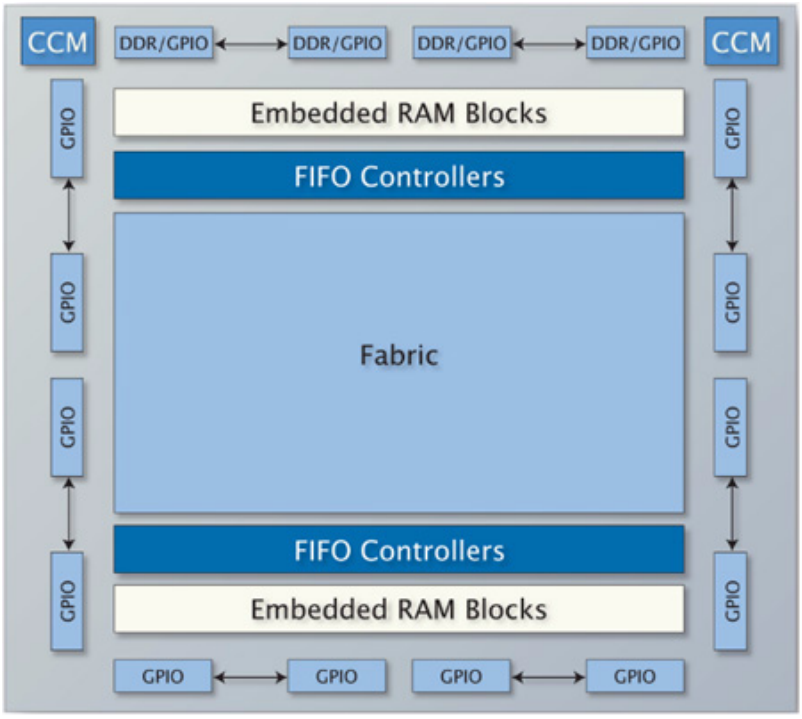
Table 6-1: Features and Benefits of the PolarPro Architecture (Continued)

Features	Benefits
CCM - Configurable Clock Manager	Flexible clock generation and manipulation eases timing closure, provides clock multiplication by 1x, 2x, and 4x as well as phase shifted clocks by 90, 180 and 270-degree phase shifts. The CCM can be used to eliminate internal device routes and external PCB trace delays, providing the ability to synchronize internal clocks to external clocks within a system. A programmable delay line enables programmable clock delays of up to 2.5 ns in programmable steps of 250 ps.
Native Support for DDR SDRAM	Native built-in DDR I/Os enable the system design to leverage the lowest cost and lowest power mobile DDR, DDR1 and DDR2 SDRAM memory.
Large Built-in FIFO/RAM blocks	Dedicated blocks of SRAM, each block is 4,608-bits and have configurable aspect ratios of 128x36, 256x18 or 512x9 (depth x width), simplifying FIFO design and implementation.

6.3 PolarPro Resources

Figure 6-1 shows the layout of the PolarPro Programmable Fabric components.

Figure 6-1: PolarPro FPGA Block Diagram

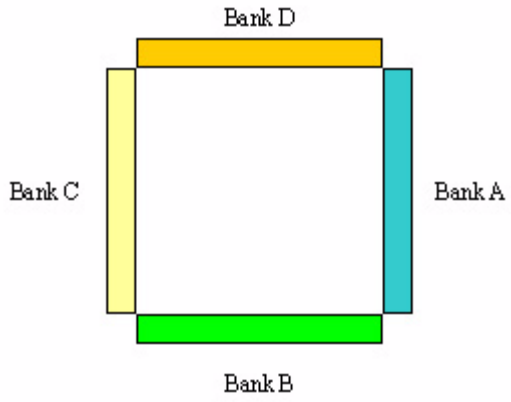


© 2008 QuickLogic Corporation

6.3.1 PolarPro I/O Banks

PolarPro I/Os are divided into separate banks. The number of banks is device dependent. Each bank of I/Os has an independent VCCIO, and therefore can support a different I/O standard within a device. **Figure 6-2** indicates the banks distribution per device and their respective maximum number of I/O per bank.

Figure 6-2: VCCIO Bank Configuration



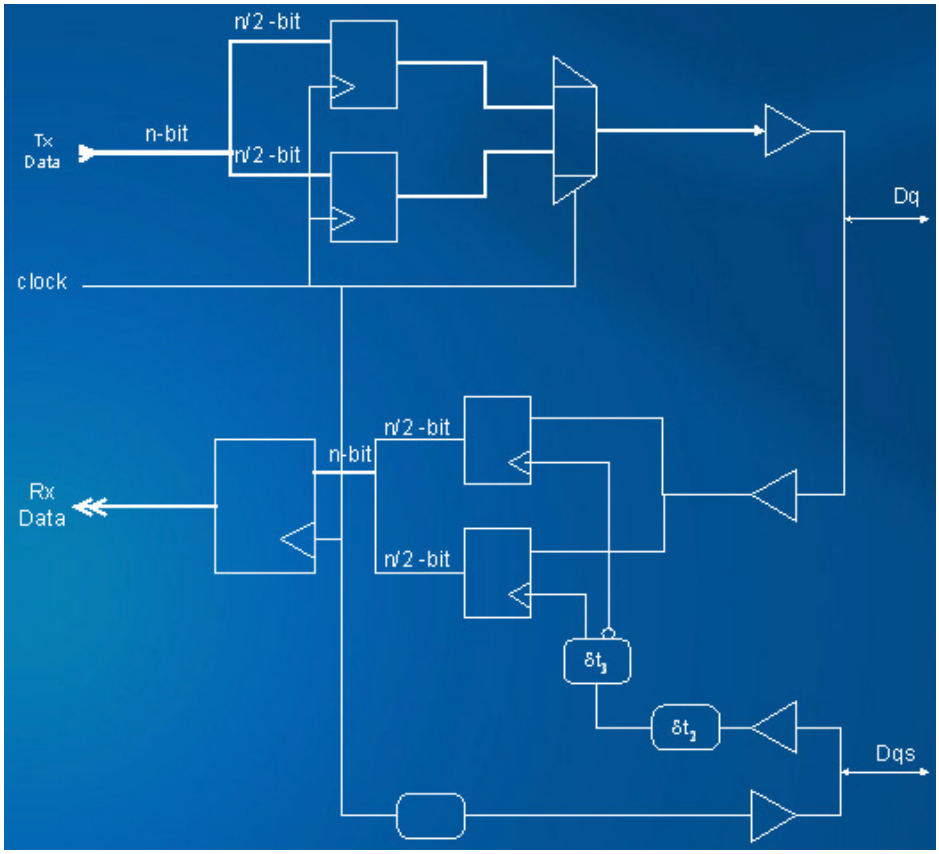
There are two types of I/Os:

- DDR I/O—located in the Bank D domain
- General Purpose I/O (GPIO)

6.3.1.1 DDRIOs

Figure 6-3 shows the DDRIO layout.

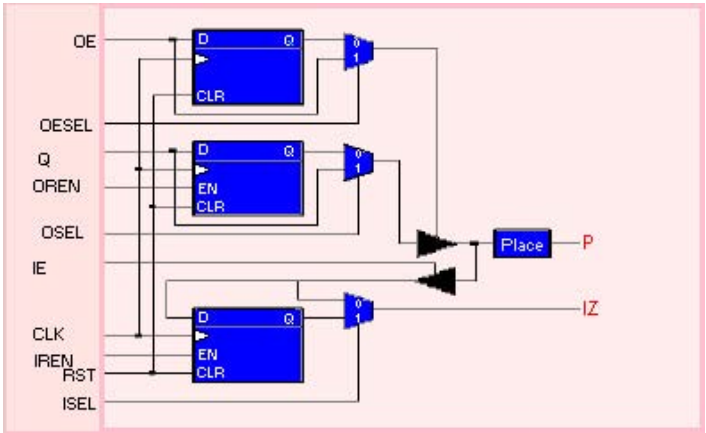
Figure 6-3: DDRIO Block Diagram



6.3.2 GPIOs

Figure 6-4 shows the GPIOs layout.

Figure 6-4: GPIO Block Diagram



© 2008 QuickLogic Corporation

The GPIO has the following features:

- Programmable slew rate control
- Programmable drive strength
- Programmable weak-keeper for pull-up/pull-down.
- Input buffer can be disabled if not needed to conserve power
- Support LVCMOS/LVTTL/3.3 V PCI
- Operating from 1.8 V to 3.3 V

6.3.3 Clock Network

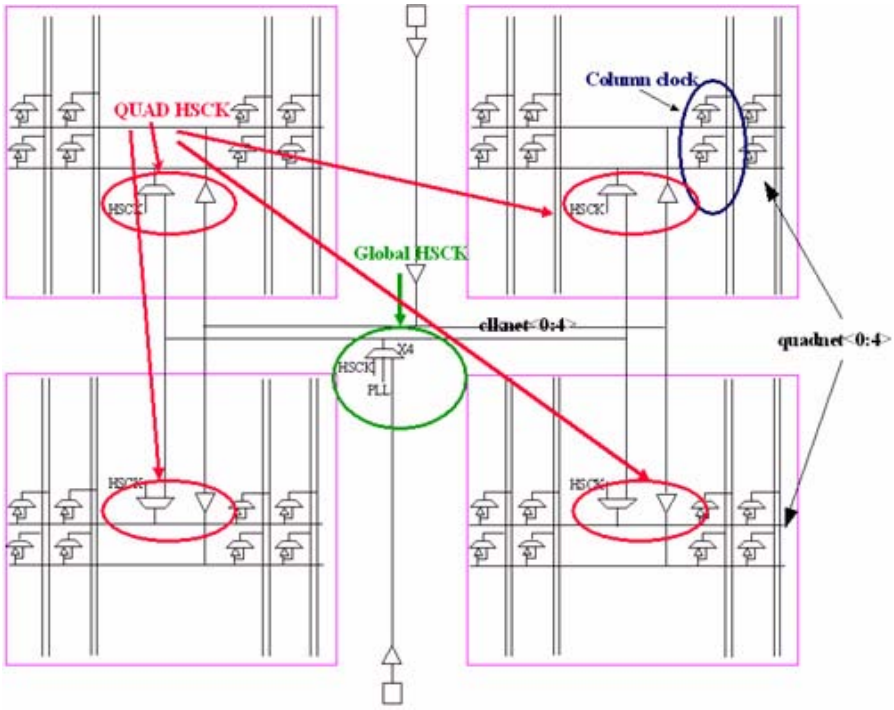
There are a total of five clock pads in all PolarPro devices. There are four Configurable Clock Manager (CCM) outputs for the middle and large device of the PolarPro family. Since there are four CCM outputs, only four of the five clocks connect through the clock/HSCK mux. The other clock connects directly to the clock network and can be used as a fast clock.

There are two levels of the clock/HSCK mux:

- Global HSCK mux—located in the middle of the die, it helps to reduce clock skew. It has three inputs: clock pin, PLL output, and global level HSCK.
- Quad level HSCK mux—located in the middle of each quad, it provides flexibility. It has two inputs: output from global HSCK mux and quad level HSCK.

The quad clock/HSCK mux for all devices has two inputs: output from the global clock mux and quad level HSCK. All clocks are inverted at the column level to provide a clock or inverted-clock option. All clocks can also be disabled at the column level.

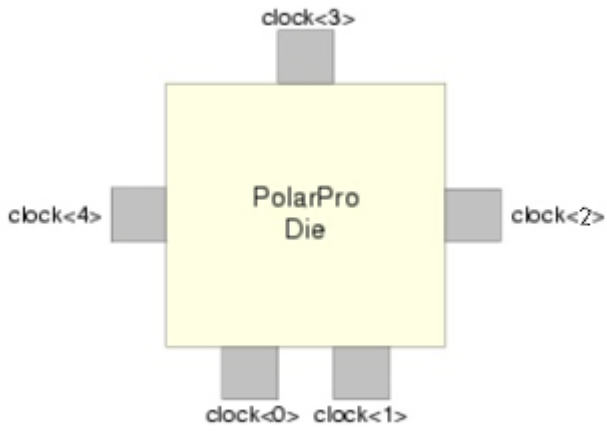
Figure 6-5: PolarPro Clock Diagram



There are five clock pads, as shown in **Figure 6-6**:

- Clock <0,1,2,4> is connected to the global HSCK mux in the middle of the device.
- Clock<2> and clock<4> are the dedicated clock inputs to the PLL.
- Clock<3> bypasses the global HSCK mux and feeds into the quadrant clock network (aka *fast clock*)

Figure 6-6: PolarPro Clock Pads



6.3.4 CCM

Flexible clock generation and manipulation eases timing closure, provides clock multiplication by 1x, 2x, and 4x as well as phase shifted clocks by 90, 180 and 270-degree phase shifts. The CCM can be used to eliminate internal device routes and external PCB trace delays, providing the ability to synchronize internal clocks to external clocks within a system. A programmable delay line enables programmable clock delays of up to 2.5 ns in programmable steps of 250 ps. **Table 6-2** lists the port definitions of the PolarPro CCM.

There are two CCMs in PolarPro devices, located in the top left and top right corners of the device. Each CCM has the choice of a dedicated clock input. The same is true for the CCM feedback. The selection of which clock input or clock feedback to use is controlled by fin_cnt and fd_cnt.

Table 6-2: CCM Port Definition

Name	Port Type	Pad / Pin	Programmable	Comment
Vdda	Power	Yes	No	Analog Vdd for PLL
Gnda	Power	Yes	No	Analog Gnd for PLL
ded_in	Input	Yes (clock 2 & 4)	No	Dedicated clock input to PLL
fin_cnt	Input	No	Yes	Clock input select
ded_fd	Input	No	No	Dedicated clock feedback
fd_cnt	Input	No	Yes	Clock feedback select to PLL
s0, s1	Input	No	Yes	Mode select for PLL
fc0, fc1	Input	No	Yes	Phase select for PLL output <1>

© 2008 QuickLogic Corporation

Table 6-2: CCM Port Definition (Continued)

Name	Port Type	Pad / Pin	Programmable	Comment
tdctl0:tdctl3	Input	No	Yes	Programmable delay control for PLL output <1>
pll_rst	Input	No	Yes	PLL reset / PLL power down mode
lock_out	Output	No	Yes	PLL lock output
Pllout0	Output	No	Yes	PLL output <0>
Pllout1	Output	No	Yes	PLL output <1>

Clock<2> is connected to the dedicated clock input of PLL1, located in the top-right corner of the BR device, while Clock<4> is connected to the dedicated clock input of PLL0, located in the top-left corner of the BR device (see **Figure 6-7** and **Table 6-3**).

Figure 6-7: Clock to PLL Connections

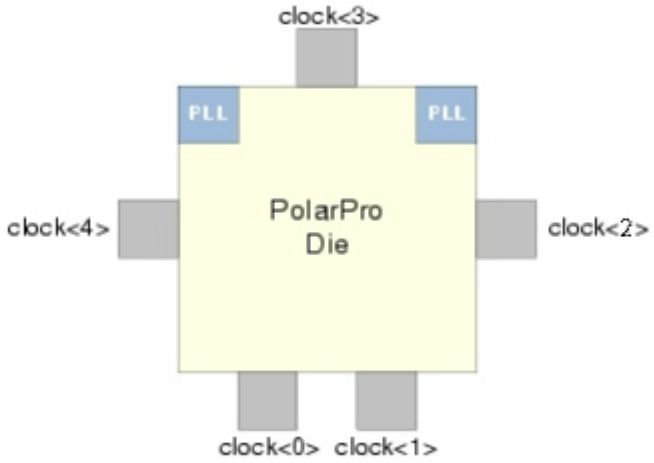


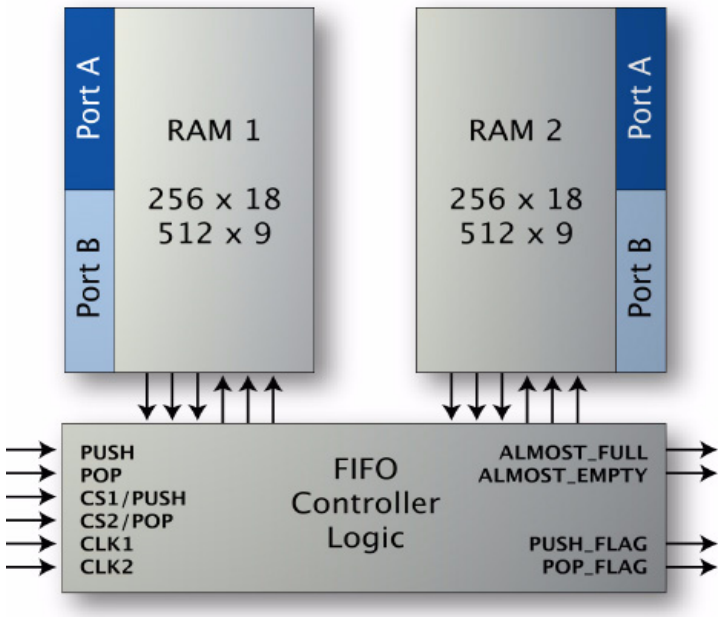
Table 6-3: Dedicated Clock Input and Feedback

PLL	Dedicated Clock Input	Dedicated Clock Feedback
0	4	colclk<4> of left interface
1	2	colclk<2> of right interface

6.3.5 RAM/FIFO

Native built-in DDR I/Os enable the system design to leverage the lowest cost and lowest power mobile DDR, DDR1 and DDR2 SDRAM memory. As shown in **Figure 6-8**, dedicated SRAM blocks are 4,608-bits long and have configurable aspect ratios of 128x36, 256x18 or 512x9 (depth x width), simplifying FIFO design and implementation.

Figure 6-8: Embedded SRAM Architecture



6.3.5.1 RAM

PolarPro devices contain embedded memory (RAM) blocks. Each RAM block has 4,608 (4 K) bits of storage. Two such blocks can be combined (internally) to get up to 8 Kbits. **Table 6-4** lists possible combination of sizes that are supported for devices with 4 Kbit RAM blocks.

Table 6-4: RAM Configuration Options

RAM Depth (bits)	RAM Width (bits)	Total bits	Required RAM Blocks
512	Up to 9	4 K	1
256	Up to 18	4 K	1
1K	Up to 9	8 K	2
512	Up to 18	8 K	2
256	Up to 36	8 K	2

The following are the main features of the possible RAM configurations listed in **Table 6-4**:

- True dual-port capability (Read/Write capability on two independent ports) in x9 and x18 configuration for 8 K RAM blocks.
- For memory utilizing one RAM block, programmable aspect ratios of 256x18 or 512x9 are independently configurable on the input and output ports.

© 2008 QuickLogic Corporation

- For memory utilizing two RAM blocks, programmable aspect ratios of 256x36, 512x18, or 1024x9 are independently configurable on the input and output ports.
- Asynchronous input and output ports – these ports can be clocked at different frequencies.
- Write enable for each byte on the data bus.

NOTE: Deeper wide bus RAM can also be achieved by concatenating the RAM blocks externally, as long as the total number of RAM blocks is not exceeded for the given device.

6.3.5.2 FIFO Controller

The PolarPro device has embedded RAM blocks and FIFO controllers. Both of these functions are implemented as hard-wired standard cell (ASIC) gates. However, their functionality is quite flexible. Each RAM block contains 4,608 bits of storage, and even in the smallest PolarPro device, there are eight of these RAM blocks. Likewise, for each of the RAM blocks there is a corresponding FIFO Controller. **Table 6-5** lists the possible FIFO configurations.

Table 6-5: FIFO Configuration Options

RAM Depth (bits)	RAM Width (bits)	Total bits	Required FIFO Controllers	Required RAM Blocks
512	Up to 9	4 K	1	1
256	Up to 18	4 K	1	1
1K	Up to 9	8 K	2	2
512	Up to 18	8 K	2	2
256	Up to 36	8 K	2	2

NOTE: Any mixture of these FIFO configurations can be implemented in a PolarPro device using multiple instances, as long as the total number of RAM blocks and FIFO controllers is not exceeded for the given device.

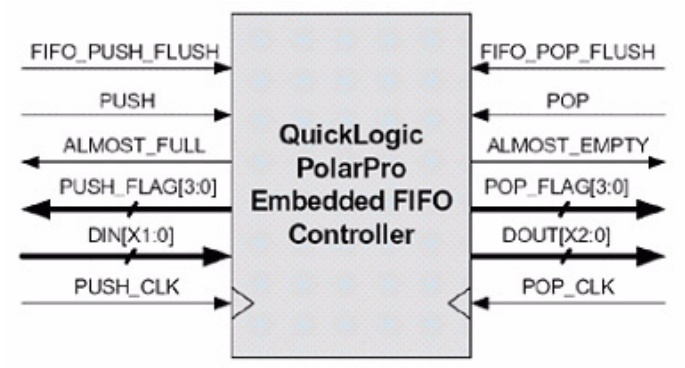
The following are the main features of the possible FIFO configurations listed in **Table 6-5**:

- Asynchronous input and output ports (these ports can be clocked at completely different frequencies).
- Almost empty and almost full output flags.
- Level indicator flag vectors for the input and output sides of the FIFO.
- Data flush inputs for the input and output sides of the FIFO.
- For FIFOs utilizing one RAM block, programmable aspect ratios of 256x18 or 512x9 are independently configurable on the input and output ports.
- For FIFOs utilizing two RAM blocks, programmable aspect ratios of 256x36, 512x18, or 1024x9 are independently configurable on the input and output ports.

© 2008 QuickLogic Corporation

Table 6-9 shows the input and output signals of the PolarPro FIFO Controller.

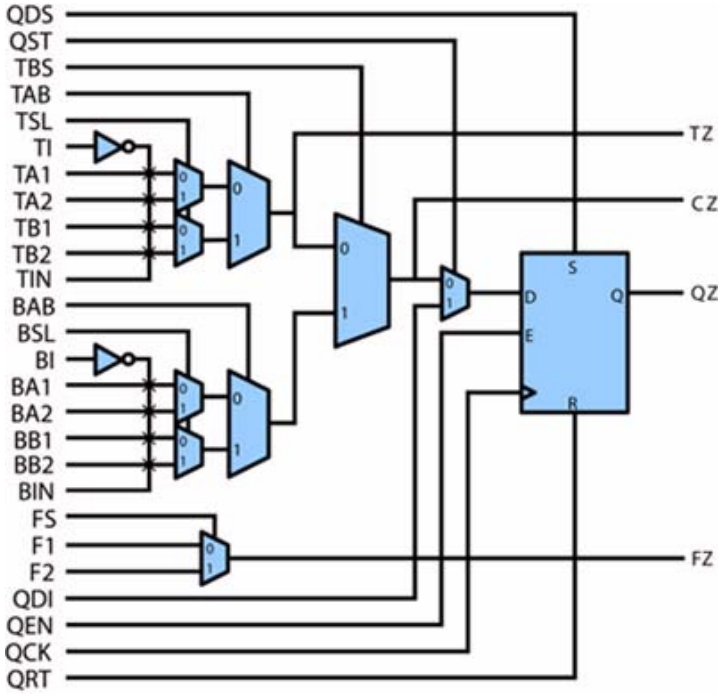
Figure 6-9: PolarPro FIFO Controller



6.3.6 Logic Cells

The PolarPro logic cell architecture is a single register, multiplexer-based logic cell designed for wide fan-in and multiple, simultaneous output functions. The high logic capacity and fan-in of the logic cell accommodates many user functions with a single level of logic delay. The PolarPro cell architecture is logically equivalent to two 3-input LUT blocks or one 4-input LUT block. The register is similar to an Enable D-Type Flip Flop with Asynchronous set and reset inputs. **Figure 6-10** shows the PolarPro logic cell architecture.

Figure 6-10: PolarPro Logic Cell



© 2008 QuickLogic Corporation

Chapter 7
PolarPro II Devices

This chapter includes the following topics:

- **"PolarPro II Overview" on page 93**
- **"Features and Benefits" on page 93**
- **"PolarPro II Resources" on page 94**

7.1 PolarPro II Overview

The PolarPro II family continues the innovation of ultra-low power in a programmable platform that is specifically designed to meet the connectivity and system logic requirements of portable and mobile applications. Building on the award winning PolarPro logic cell architecture, PolarPro II extends QuickLogic's leadership in low power architecture with an advanced power management scheme through the enhanced Very Low Power (VLP) mode and voltage scaling. An optimized I/O structure allows it to fit in BGA packages as small as 5 mm x 5 mm, reducing system BOM costs and PCB area. Eight I/O banks and six RAM blocks with varying densities provide designers with higher design flexibility and efficiency than the previous generation architectures.

7.2 Features and Benefits

Table 7-1 describes the main features of the PolarPro II device family architecture.

Table 7-1: Features and Benefits of the PolarPro II Architecture

Features	Benefits
Enhanced Logic Cell Architecture	A flexible, synthesis friendly logic cell that enables efficient mapping of up to 13-bit wide input functions, as well as mapping of any 4-input LUT, or two 3-input LUT combinations into a single level of logic. Each logic cell has four simultaneous outputs and a dedicated Enable D-Type Flip Flop.
VLP Mode	From an external input control pin, the FPGA device can be put into Very Low Power (VLP) mode, in which the device will typically draw less than 10 µA. Within the VLP mode, I/O states and internal register values are retained. This capability provides an instant ability to save battery power when the device function is not needed. VLP mode operates at 1.5 V and 1.8 V.
Flexible Clock Networks	There are five clocks in each of the four quadrants. This allows for clocking on one or more quadrants, or on a global clock basis, resulting in a more efficient use of resources. All clocks can be dynamically disabled at the column level/pad level also. The dynamic clock enable/disable feature at the pad input provides clock de-glitching during VLP mode transitions.

© 2008 QuickLogic Corporation

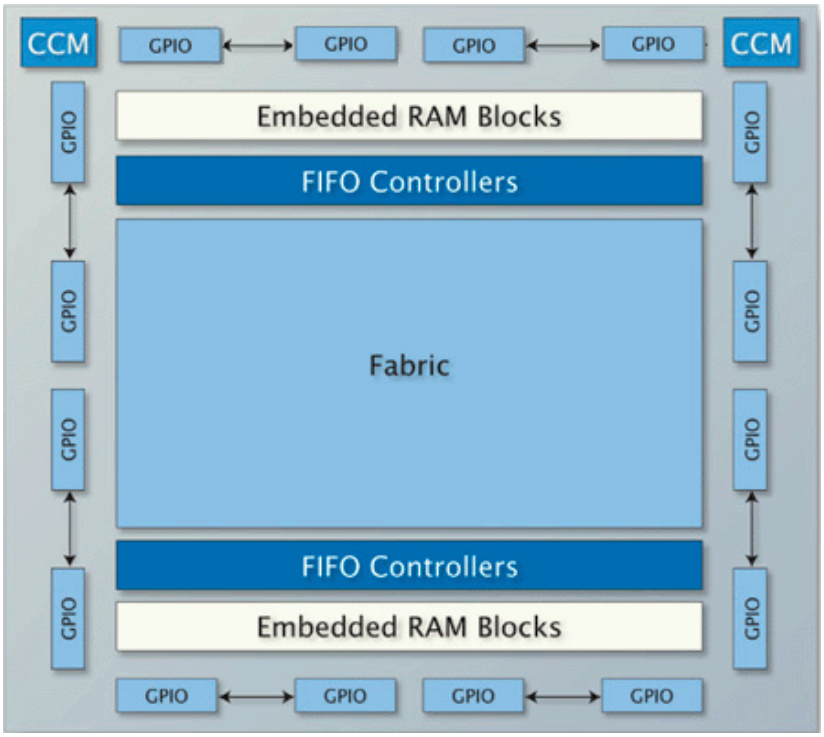
Table 7-1: Features and Benefits of the PolarPro II Architecture (Continued)

Features	Benefits
Programmable I/Os	General purpose programmable I/Os with fully registered Input/Outputs. Each I/O has a programmable Slew Rate control and programmable Pull-Down functionality. Supports LVTTL and LVCMOS up to 200 MHz and PCI 2.3 up to 66 MHz.
CCM - Configurable Clock Manager	Flexible clock generation and manipulation eases timing closure, provides clock multiplication by 1x, 2x, and 4x as well as phase shifted clocks by 90, 180 and 270-degree phase shifts. The CCM can be used to eliminate internal device routes and external PCB trace delays, providing the ability to synchronize internal clocks to external clocks within a system. A programmable delay line enables programmable clock delays of up to 2.5 ns in programmable steps of 250 ps.
Large Built-in FIFO/RAM blocks	Dedicated blocks of SRAM. Each block is 4,608-bits or 2,304-bits and have configurable aspect ratios of 256x18 or 512x9 (depth x width) for 4 K and 128x18 or 256x9 for 2 K, simplifying FIFO design and implementation. They can be configured to function as dual-port or FIFO RAM memories. There are two types of RAM blocks: • RAM_Block_2K, consisting of a 2-Kbit RAM memory • RAM_Block_4K, consisting of a 4-Kbit RAM memory

7.3 PolarPro II Resources

Figure 7-1 shows the layout of the PolarPro II Programmable Fabric components.

Figure 7-1: PolarPro II FPGA Block Diagram

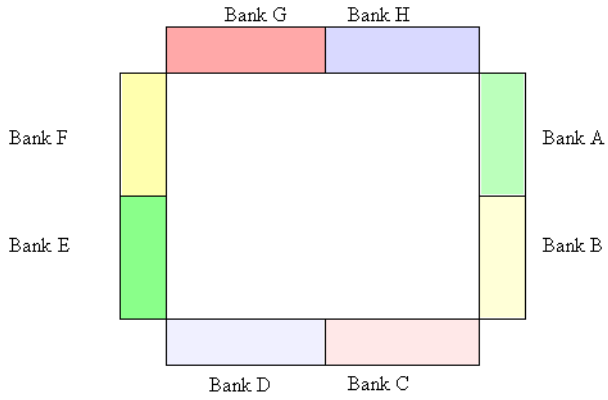


© 2008 QuickLogic Corporation

7.3.1 PolarPro II I/O Banks

PolarPro II I/Os are divided into separate banks. Each bank of I/Os has an independent VCCIO, and therefore can support a different I/O standard within a device. **Figure 7-2** indicates the banks distribution per device.

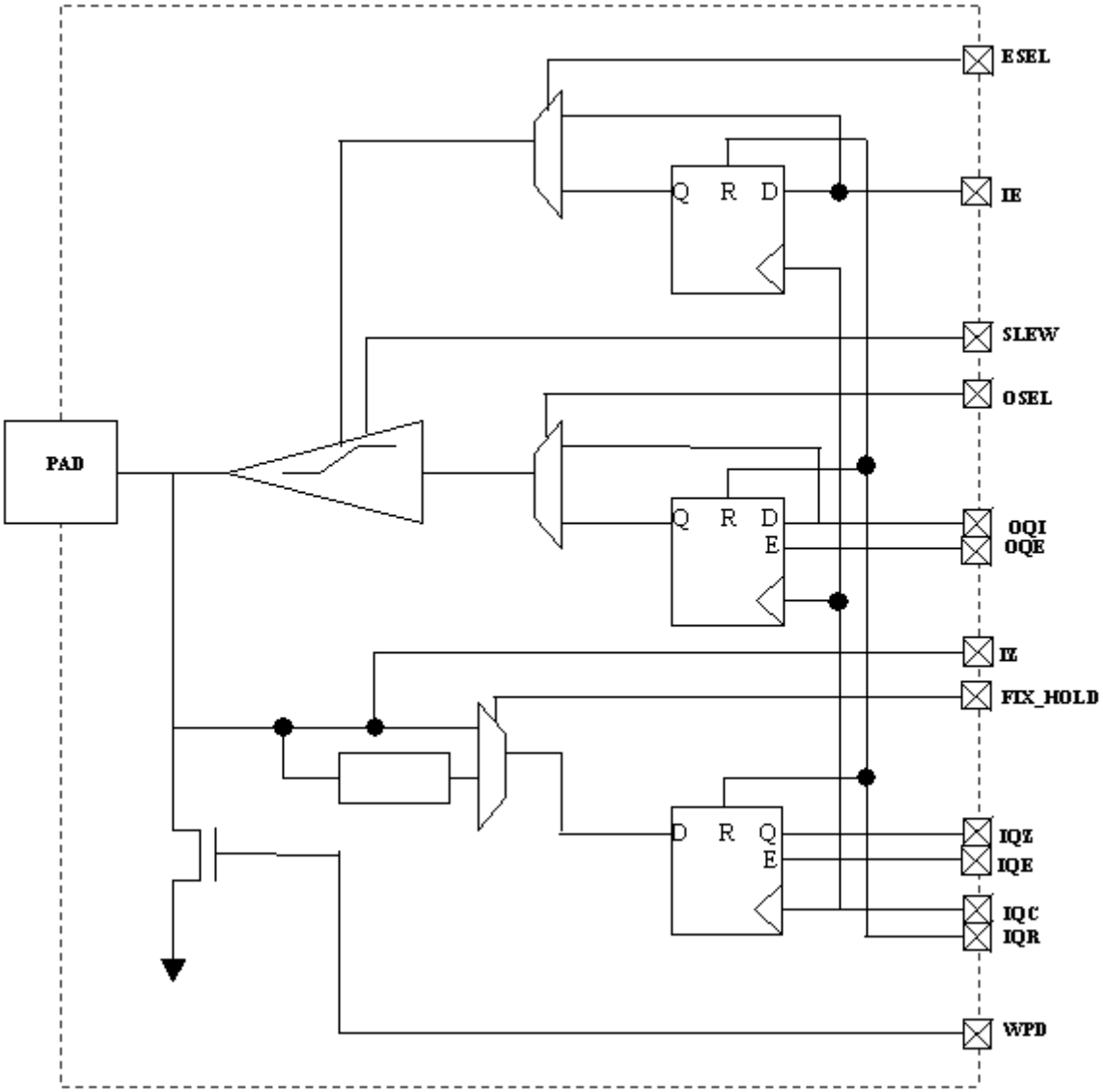
Figure 7-2: VCCIO Bank Configuration



7.3.2 GPIOs

Figure 7-3 shows the GPIOs layout.

Figure 7-3: GPIO Block Diagram



The GPIO has the following features:

- Programmable slew rate control
- Supports one drive strength only at any given VCCIO
- FIX_HOLD function that provides a way to improve the hold time of a design
- Concurrent combinatorial and register input provides edge detection circuit
- Data retention
- Programmable weak pull-down
- Supports LVCMOS/LVTTL/3.3 V PCI
- Operates from 1.8 V to 3.3 V

© 2008 QuickLogic Corporation

7.3.3 Clock Network

There are a total of five clock pads in all PolarPro II devices. When the clocks are not used, they can be used as GPIOs. There are two Configurable Clock Manager (CCM) outputs for the PolarPro II device. Since there are two CCM outputs, only four of the five clocks connect through the global clock/HSCK mux. The other clock connects directly to the clock network and can be used as a fast clock.

There are two levels of the clock/HSCK mux:

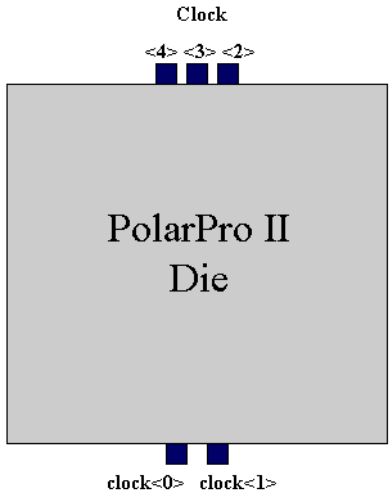
- Quad level HSCK mux—located in the middle of each quad, it provides flexibility. It has four inputs: inverted and noninverted output from global HSCK mux and quad level HSCK.
- Global HSCK mux—located in the middle of the die, it helps to reduce clock skew. It has three inputs: clock pin, PLL output, and global level HSCK.

All clocks can be disabled at the column level/pad level also. Dynamic Clock enable/disable feature at pad input provides clock de-glitching during VLP mode transitions.

There are five clock pads, as shown in **Figure 7-4**:

- Clock <0,1,2,4> is connected to the global HSCK mux in the middle of the device.
- Clock<2> is the clock input to the PLL.
- Clock<3> bypasses the global HSCK mux and feeds into the quadrant clock network (aka *fast clock*)

Figure 7-4: PolarPro II Clock Pads



7.3.4 CCM

Flexible clock generation and manipulation eases timing closure, provides clock multiplication by 1x, 2x, and 4x as well as phase shifted clocks by 90, 180 and 270-degree phase shifts. The CCM can be used to eliminate internal device routes and external PCB trace delays, providing the ability to synchronize internal clocks to external clocks within a system. A programmable delay line enables programmable clock delays of up to 2.5 ns in programmable steps of 250 ps. **Table 7-2** lists the port definitions of the PolarPro II CCM.

© 2008 QuickLogic Corporation

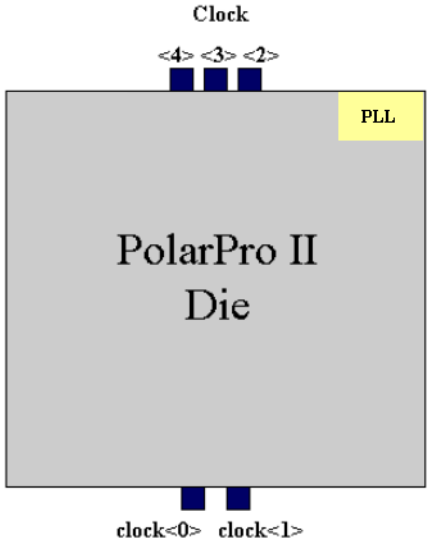
There is one CCM in the PolarPro II device, located in the top right corner of the device. The CCM has the choice of a dedicated clock input and feedback. The selection of the clock input or clock feedback to use is controlled by fin_cnt and fd_cnt.

Table 7-2: CCM Port Definition

Name	Port Type	Pad / Pin	Programmable	Comment
Vdda	Power	Yes	No	Analog Vdd for PLL
Gnda	Power	Yes	No	Analog Gnd for PLL
ded_in	Input	Yes (clock 2)	No	Dedicated clock input to PLL
fin_cnt	Input	No	Yes	Clock input select
ded_fd	Input	No	No	Dedicated clock feedback
fd_cnt	Input	No	Yes	Clock feedback select to PLL
s0, s1	Input	No	Yes	Mode select for PLL
fc0, fc1	Input	No	Yes	Phase select for PLL output <1>
tdctl0:tdctl3	Input	No	Yes	Programmable delay control for PLL output <1>
pll_rst	Input	No	Yes	PLL reset / PLL power down mode
lock_out	Output	No	Yes	PLL lock output
Pllout0	Output	No	Yes	PLL output <0>
Pllout1	Output	No	Yes	PLL output <1>

Clock<2> is connected to the dedicated clock input of PLL0, located in the top right corner of the QLPP2BR device (see **Figure 7-5** and **Table 7-3**).

Figure 7-5: Clock to PLL Connections



© 2008 QuickLogic Corporation

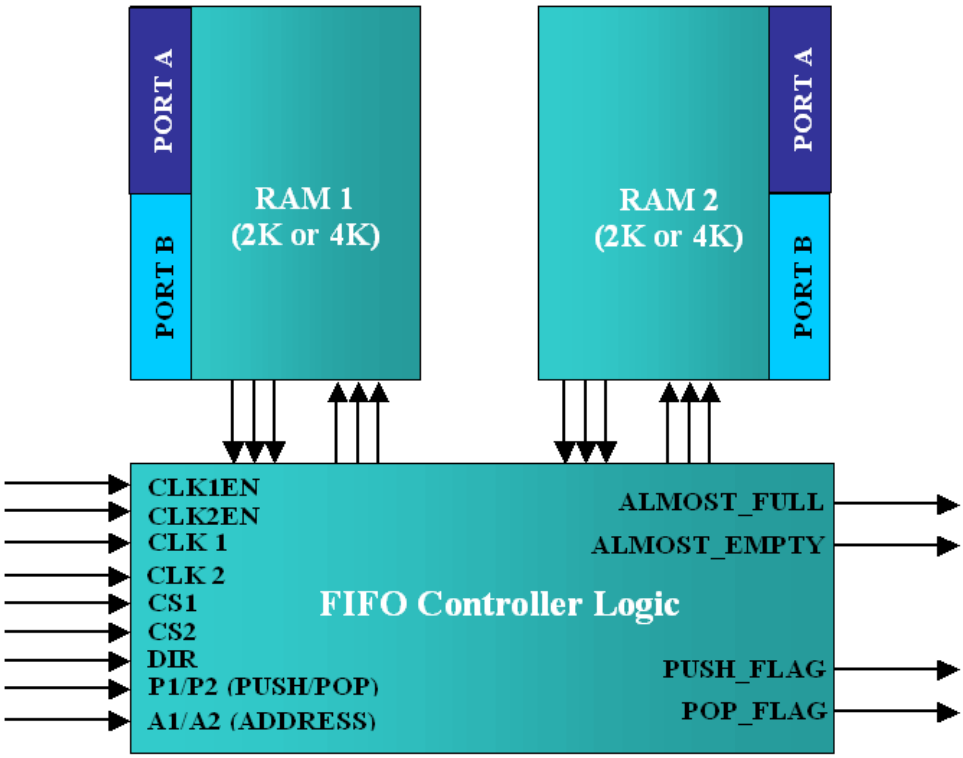
Table 7-3: Dedicated Clock Input and Feedback

PLL	Dedicated Clock Input	Dedicated Clock Feedback
0	2	colclk<2> of right interface

7.3.5 RAM/FIFO

The RAM blocks in the PolarPro II device enable designers to implement various types of memories in the FPGA. They can be configured to function as dual-port or FIFO RAM memories. As shown in **Figure 7-6**, dedicated SRAM blocks are 2,304 bits or 4,608 bits long and have configurable aspect ratios of 256x18 or 512x9 (depth x width) for 4 K and 128x18 or 256x9 for 2 K, simplifying FIFO design and implementation.

Figure 7-6: Embedded SRAM Architecture



7.3.5.1 RAM

PolarPro II devices contain embedded memory (RAM) blocks. Each RAM block has 4,608 (4 K) bits or 2,304 (2 K) bits of storage. Two 4 K blocks can be combined (internally) to get up to 8 Kbits, and two 2 K blocks can be combined to get up to 4 Kbits. The RAM blocks inside the PolarPro II devices are configurable as dual-port or as FIFO memories.

There are two types of RAM blocks:

- RAM_Block_2K, consisting of a 2-Kbit RAM memory
- RAM_Block_4K, consisting of a 4-Kbit RAM memory

© 2008 QuickLogic Corporation

The top RAM block is identical to PolarPro 4 K RAM blocks. The bottom RAM block is half the density of the top RAM block.

Table 7-4 lists possible combination of sizes that are supported for devices with 4 Kbit and 2 Kbit RAM blocks.

Table 7-4: RAM Configuration Options

RAM Depth (bits)	RAM Width (bits)	Total bits	Required RAM Blocks
512	Up to 9	4 K	1
256	Up to 18	4 K	1
1K	Up to 9	8 K	2
512	Up to 18	8 K	2
256	Up to 36	8 K	2
256	Up to9	2 K	1
128	Up to 18	2 K	1
512	Up to 9	4 K	2
256	Up to 18	4 K	2
128	Up to 36	4 K	2

The following are the main features of the possible RAM configurations listed in **Table 7-4**:

- True dual-port capability (Read/Write capability on two independent ports) in x9 and x18 configuration for 8 K RAM blocks.
- For memory utilizing one 4 K RAM block, programmable aspect ratios of 256x18 or 512x9 are independently configurable on the input and output ports.
- For memory utilizing one 2 K RAM block, programmable aspect ratios of 256x9 or 128x18 are independently configurable on the input and output ports.
- For memory utilizing two 4 K RAM blocks, programmable aspect ratios of 256x36, 512x18, or 1024x9 are independently configurable on the input and output ports.
- For memory utilizing two 2 K RAM blocks, programmable aspect ratios of 128x36, 556x18, or 512x9 are independently configurable on the input and output ports.
- Asynchronous input and output ports – these ports can be clocked at different frequencies.
- Write enable for each byte on the data bus.
- Clock disabling during idle operation

NOTE: Deeper wide bus RAM can also be achieved by concatenating the RAM blocks externally, as long as the total number of RAM blocks is not exceeded for the given device.

7.3.5.2 FIFO Controller

The PolarPro II device has embedded RAM blocks and FIFO controllers. Both of these functions are implemented as hard-wired standard cell (ASIC) gates. However, their functionality is quite flexible. Each RAM block contains 4,608 bits or 2,304 bits of storage, and even in the smallest PolarPro II device, there are eight of these RAM blocks. Likewise, for each of the RAM blocks there is a corresponding FIFO Controller. **Table 7-5** lists the possible FIFO configurations.

© 2008 QuickLogic Corporation

Table 7-5: FIFO Configuration Options

RAM Depth (bits)	RAM Width (bits)	Total bits	Required FIFO Controllers	Required RAM Blocks
512	Up to 9	4 K	1	1
256	Up to 18	4 K	1	1
1K	Up to 9	8 K	2	2
512	Up to 18	8 K	2	2
256	Up to 36	8 K	2	2
256	Up to 9	2 K	1	1
128	Up to 18	2 K	1	1
512	Up to 9	4 K	2	2
256	Up to 18	4 K	2	2
128	Up to 36	4 K	2	2

NOTE: Any mixture of these FIFO configurations can be implemented in a PolarPro II device using multiple instances, as long as the total number of RAM blocks and FIFO controllers is not exceeded for the given device.

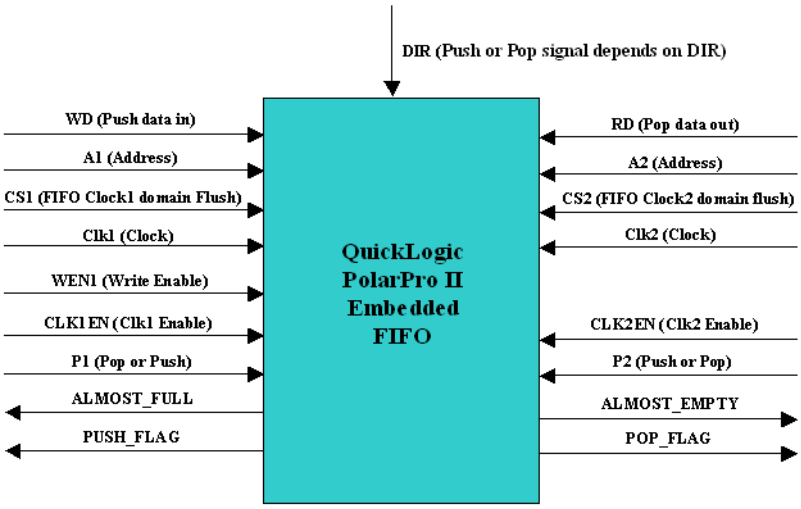
The following are the main features of the possible FIFO configurations listed in **Table 7-5**:

- Asynchronous input and output ports (these ports can be clocked at completely different frequencies).
- Almost empty and almost full output flags.
- Level indicator flag vectors for the input and output sides of the FIFO.
- Data flush inputs for the input and output sides of the FIFO.
- For FIFOs utilizing one 4 K RAM block, programmable aspect ratios of 256x18 or 512x9 are independently configurable on the input and output ports.
- For FIFOs utilizing one 2 K RAM block, programmable aspect ratios of 256x9 or 128x18 are independently configurable on the input and output ports.
- For FIFOs utilizing two 4 K RAM blocks, programmable aspect ratios of 256x36, 512x18, or 1024x9 are independently configurable on the input and output ports.
- For FIFOs utilizing two 2 K RAM blocks, programmable aspect ratios of 128x36, 512x18, or 512x9 are independently configurable on the input and output ports.
- Clock disabling during idle operation.
- Asynchronous reset.
- Switchable clock domain between PUSH and POP side during asynchronous operation.

© 2008 QuickLogic Corporation

Figure 7-7 shows the input and output signals of the PolarPro II FIFO Controller.

Figure 7-7: PolarPro II FIFO Controller

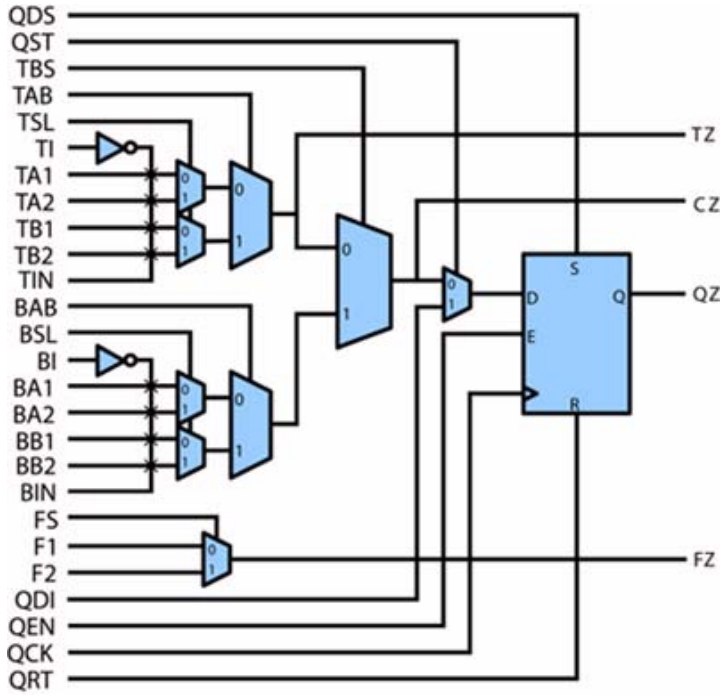


7.3.6 Logic Cells

The PolarPro II logic cell architecture is a single register, multiplexer-based logic cell designed for wide fan-in and multiple, simultaneous output functions. The high logic capacity and fan-in of the logic cell accommodates many user functions with a single level of logic delay. The PolarPro II cell architecture is logically equivalent to two 3-input LUT blocks or one 4-input LUT block. The register is similar to an Enable D-Type Flip Flop with Asynchronous set and reset inputs. **Figure 7-8** shows the PolarPro II logic cell architecture.

© 2008 QuickLogic Corporation

Figure 7-8: PolarPro II Logic Cell



© 2008 QuickLogic Corporation

Chapter 8
Power Calculator

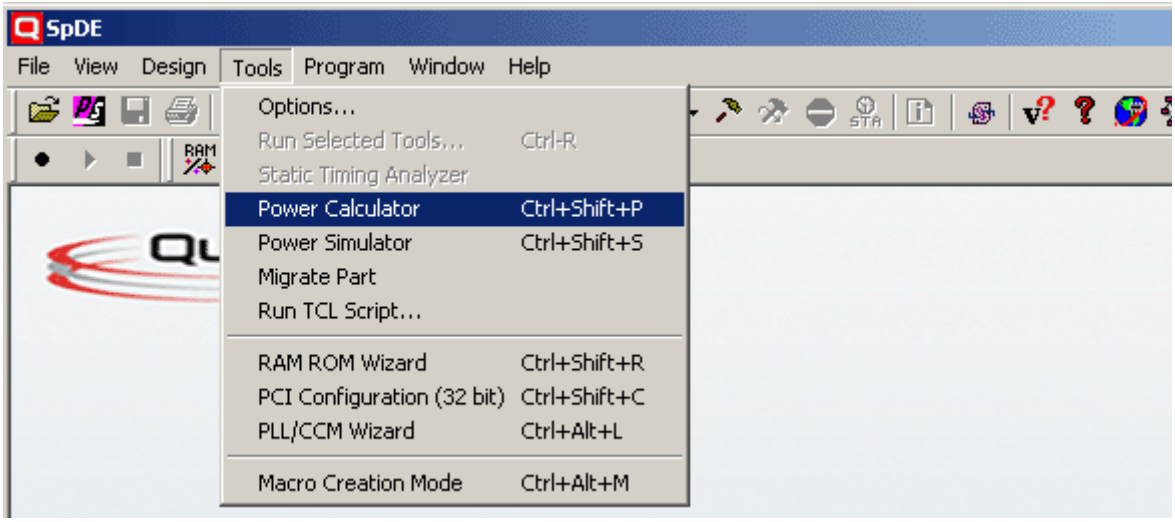
QuickWorks is equipped with an advanced Power Calculator. The Power Calculator helps to automatically calculate the approximate power consumed by your design.

This chapter contains the following sections:

- **"Using Power Calculator" on page 105**
- **"Power Consumption Parameters" on page 107**
- **"Power Calculation Assumptions" on page 110**
- **"Power Calculator Output" on page 110**
- **"TCL/CLI Support" on page 110**

8.1 Using Power Calculator

To launch the Power Calculator, click on the **Power Calculator** icon in SpDE's toolbar, or select **Tools>Power Calculator**.

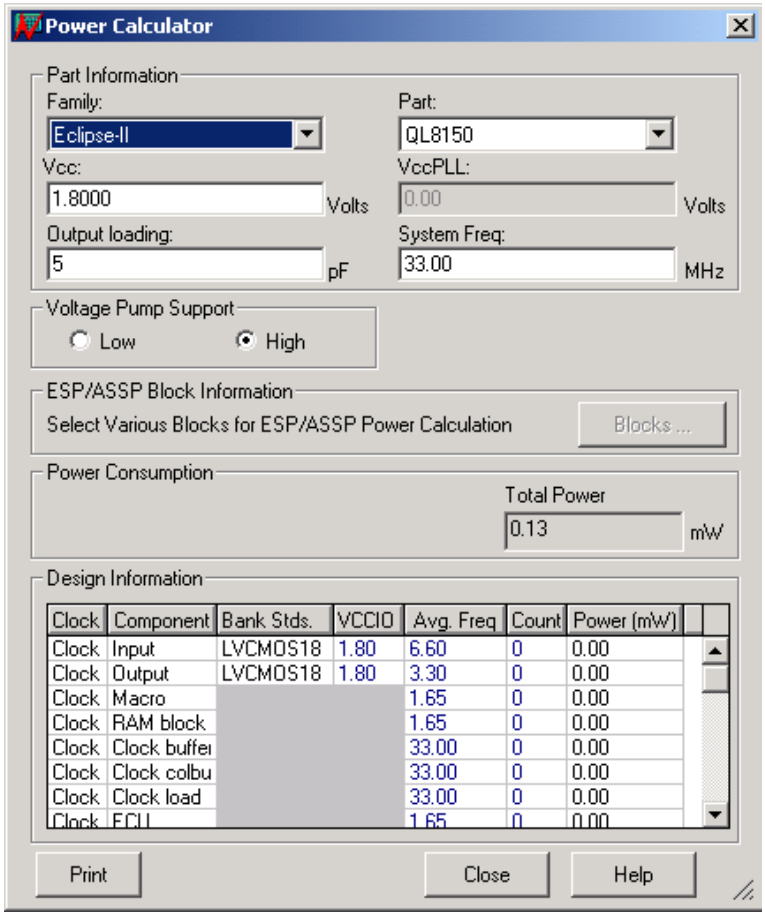


There are two unique ways how the Power Calculator calculates power:

- **The first method** is used when no design file is loaded. The user can select the desired QuickLogic device family and part from the drop-down menu. The Power Calculator shows the worst-case scenario in terms of power consumption in the selected device, assuming maximum utilization of logic cells.

© 2008 QuickLogic Corporation

- **The second method** is by opening up the `.chp` file for the corresponding design in SpDE and then opening the Power Calculator. You can also load a QDIF file and then run one or more QuickWorks tools instead of loading the `.chp` file. The Power Calculator will then compute an approximate value for the consumed power.

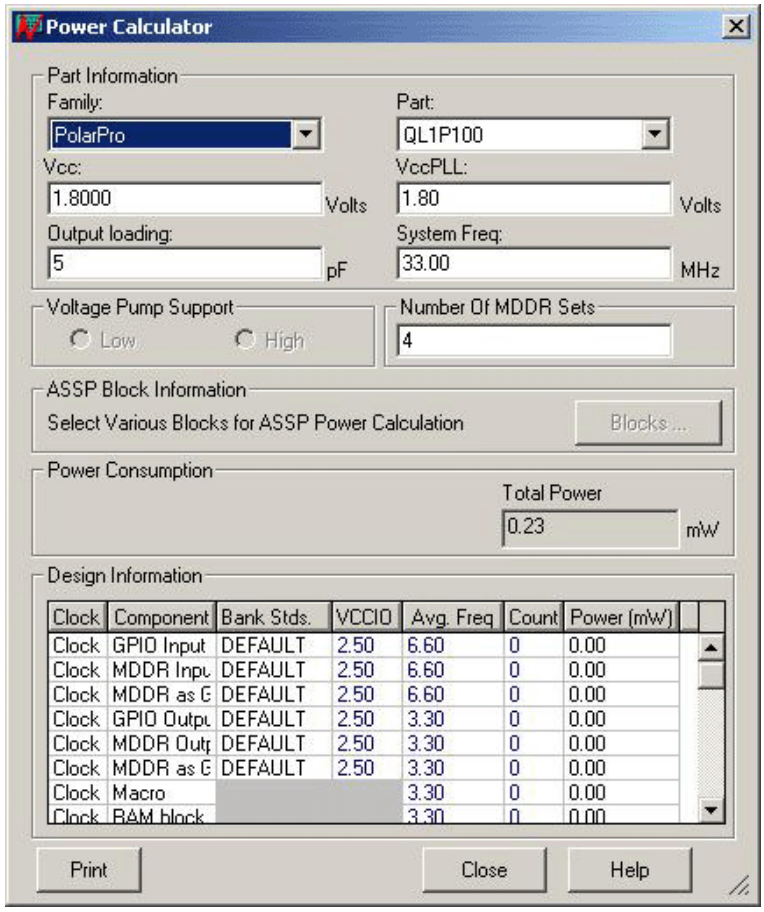

Power Calculator
Part Information
Family:
Eclipse-II
Part:
QL8150
Vcc:
1.8000
Volts
VccPLL:
0.00
Volts
Output loading:
5
pF
System Freq:
33.00
MHz
Voltage Pump Support
Low
High
ESP/ASSP Block Information
Select Various Blocks for ESP/ASSP Power Calculation
Blocks ...
Power Consumption
Total Power
0.13
mW
Design Information
Clock | Component | Bank Stds. | VCCIO | Avg. Freq | Count | Power (mW)
Clock | Input | LVCMOS18 | 1.80 | 6.60 | 0 | 0.00
Clock | Output | LVCMOS18 | 1.80 | 3.30 | 0 | 0.00
Clock | Macro | | | 1.65 | 0 | 0.00
Clock | RAM block | | | 1.65 | 0 | 0.00
Clock | Clock buffer | | | 33.00 | 0 | 0.00
Clock | Clock colbu | | | 33.00 | 0 | 0.00
Clock | Clock load | | | 33.00 | 0 | 0.00
Clock | FCU | | | 1.65 | 0 | 0.00
Print
Close
Help

© 2008 QuickLogic Corporation

8.2 Power Consumption Parameters

The Power Calculator is a tool used in estimating the power dissipated by an FPGA and an ASSP based on various parameters. For a selected part, the power calculator tool estimates the power dissipated by the design based on a set of user-specified requirements and statistical data. The tool can also estimate power for a specific design under implementation using the selected part for that design and its utilization. For more details, refer to **"FPGA Power Calculation" on page 107** and **"ASSP Block Power" on page 108**.

8.2.1 FPGA Power Calculation



The following input parameters are used for the computation of the total power consumption for the device/design:

- **Part Information**—select a family and one of the parts belonging to that family:
  - Family—device family name
  - Part—part name
  - Vcc—operating voltage or power supply pin voltage
  - VccPLL—voltage at PLL/CCM supply pin

  - Output Loading—capacitive load on outputs (default output loading is 5 pF)
  - System Freq—system clock toggle frequency (default system frequency is 33 MHz)
- **Voltage Pump Support**—Eclipse II devices support Voltage Pump selection. It is set to "High" by default because it consumes less static power in this state. It can be set to "Low", if needed. Power consumption of all Eclipse II devices can be reduced significantly by de-activating the charge pumps inside the architecture. By applying 3.3 V to the VPUMP pin, the internal charge pump is deactivated, effectively reducing the static and dynamic power consumption of the device.
- **No of MDDR Sets**—number of sets used. This input is available for the PolarPro device family only when a design is not loaded. When a design is loaded, it is displayed as a read-only value, showing how many sets are used by the design.
- **Design Information**—FPGA components and their power consumption:
  - Clock—clock name
  - Component—input or output components
  - Bank Standards—I/O bank standards. It is not applicable for pre-pinnacle devices. You can change this value if design is not loaded.
  - VCCIO—I/O supply voltage. It is not applicable for pre-pinnacle devices. You can change this value in the range of +/-10% of the default value for the selected bank standard.
  - Average Frequency—average frequency of the clock component. You can change this value. By default, it is a percentage of System Frequency, and it should not exceed the System Frequency value.
  - Count—count for the component. You can change this value. It should not exceed the resources available for selected device.
  - Power (mW)—power consumed by the components
- **ASSP Block Information**—select ASSP blocks for ASSP power calculation:
  - ASSP Block Button—opens the ASSP block power calculator.

The tool supports default Vcc and VccPLL (if supported by the selected device), which can be changed within a pre-defined range.

For the .25 µm- and .18 µm-based architectures, the *VccIO* is also supported, which can be different for each I/O *Bank Standard*. The power consumption for a selected I/O bank configuration of the loaded design is displayed under the Design Information field.

NOTE: When the design is not loaded (first method - see **"Using Power Calculator" on page 105**), you can change the I/O Bank Standards and view the estimated power, whereas when the design is loaded (second method - see **"Using Power Calculator" on page 105**), the Power Calculator does not allow you to change I/O Bank Standards. Instead, you must open the Configuration Editor, change the I/O Bank Standards information, and then open the Power Calculator to see the results.

The Power Calculator also handles cases where the design uses multiple clock trees. It calculates power by considering the clock trees independently and then showing the total power consumed. The multiple clock trees can be viewed by scrolling down the Clock column.

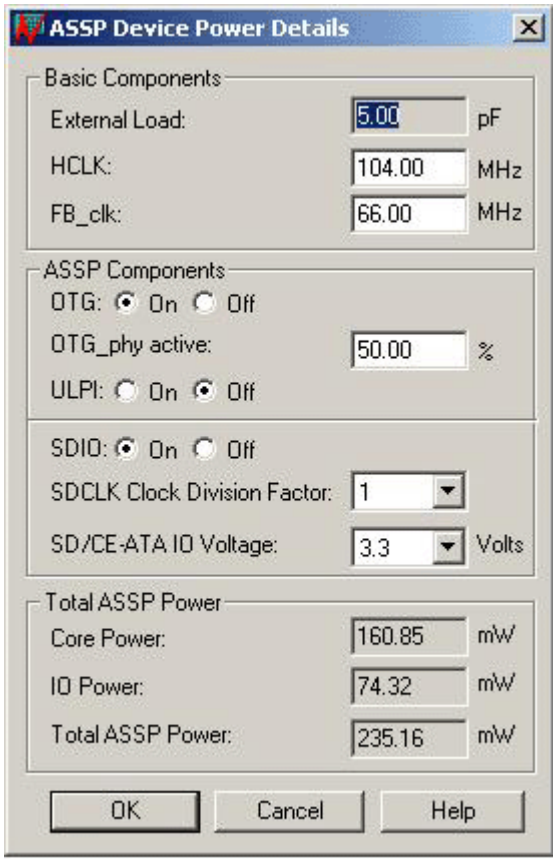
8.2.2 ASSP Block Power

When the selected part supports an ASSP, power should be estimated separately for the ASSP usage, and it should also be added to the total design power.

© 2008 QuickLogic Corporation

To get correct estimates of ASSP power, you can specify the required configuration of the ASSP blocks and their operating frequency. If a design is loaded, and it uses the ASSP block, the configuration data is extracted from the ASSP block connections to FPGA and then the power is estimated.

For example, for a QL1A100 ArcticLink device, inputs need to be defined as shown.



8.3 Power Calculation Assumptions

The Power Calculator makes the following assumptions:

- The inputs are assumed to toggle at 1/5 of the system frequency.
- The outputs are assumed to toggle at 1/10 of the system frequency.
- The macros, RAMs and ECUs are assumed to toggle at 1/20 of the system frequency.
- The clock buffers, clock column buffers, clock loads and PLLs toggle with the system frequency.

NOTE: You can modify these assumptions to better model the design. The Total Power Consumption field shows the approximate value of the total power consumed.

8.4 Power Calculator Output

The Power Calculator output can be printed using the **Print** option. In addition, the file can be saved as an Excel spreadsheet.

8.5 TCL/CLI Support

The Power Calculator can also be run as a TCL command within QuickWorks or as a command line option with the command line version of QuickWorks.

For more information, see also:

- **"POWERCALC" on page 29** - Using Command Line Power Calculator
- **"TCL" on page 30** - Using TCL Scripts
- **"PowerCalc" on page 34** - Using Power Calculator TCL Scripts
- **"Power Calculator Input File" on page 46**

© 2008 QuickLogic Corporation

Chapter 9
Power Simulator

This chapter describes the QuickLogic Power Simulator tool and contains the following sections:

- **"Functional Overview" on page 111**
- **"Power Simulator Environment Setup" on page 111**
- **"Using Power Simulator" on page 112**
- **"Limitations" on page 118**

9.1 Functional Overview

The Power Simulator GUI application works with third-party simulators, such as Active-HDL and ModelSim, to calculate the design dynamic power consumption for any given QuickLogic design. Once the total design power consumption is calculated, an output power profile result file (`powersim.log`) is generated and contains the following:

- Total power consumed by the design
- Operating voltage
- Operating frequency
- Peak power of the design

This log file is later used as an input to the Power Extrapolation tool, which generates a power extrapolation graph based on varying operating voltage and operating frequency.

9.2 Power Simulator Environment Setup

This section describes the Power Simulator environment setup, including the required inputs for power simulation to be completed successfully, the intermediate output files generated by Power Simulator, and the final output file reporting the total design power consumption, the operating voltage, the operating frequency and the peak power of the design.

The inputs to the Power Simulator tool are:

- Placed, Routed, Delay-Simulated, and Back-annotated QuickLogic design file (`.chp`)
- Encrypted `.vp/.vhp` file
- Back-annotated output files (`.vq/.vhq, .sdf`)
- Test bench files (`.tf/.tb`)—there can be multiple `.tf/.tb` files for any single design
- Power Calculator DLLs for Active-HDL and ModelSim simulators for Verilog HDL and Active-HDL simulator support for VHDL should be located at ..\pasic\spde or ..\<Current Design Directory>

© 2008 QuickLogic Corporation

- Power Simulator templates to create the macro file for Active-HDL and ModelSim simulators for Verilog HDL and VHDL flow should be located at ..\pasic\spde or ..\<Current Design Directory>

The intermediate files and the final result output file generated by Power Simulator are:

- Intermediate output files:
  - The simulation task file for the design (`.mtf/.mtb`) at ..\<Current Design Directory>
  - Power Simulator macro file (`qlPowerCalc.do`) at ..\<Current Design Directory>
- Final output file
  - Output power profile result file (`powersim.log`)

9.3 Using Power Simulator

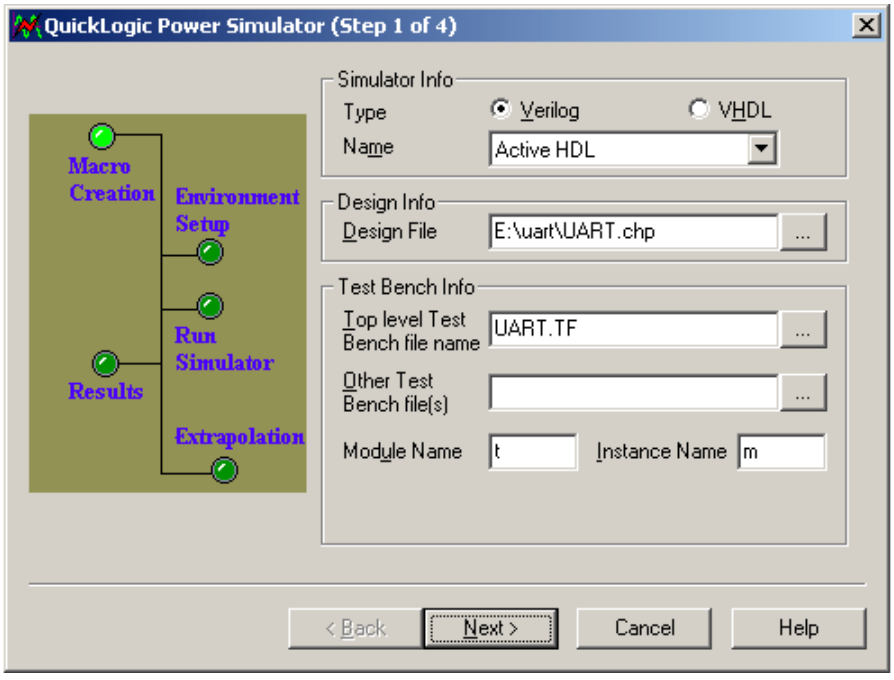
9.3.1 Invoking Power Simulator Tool from SpDE

The Power Simulator can be invoked from SpDE using:

- Toolbar—from the drop-down menu of the **Power Calculator** toolbar icon.
- Menu—by selecting **Tools>Power Simulator**.
- Shortcut key—by pressing **Ctrl + Shift + S**.

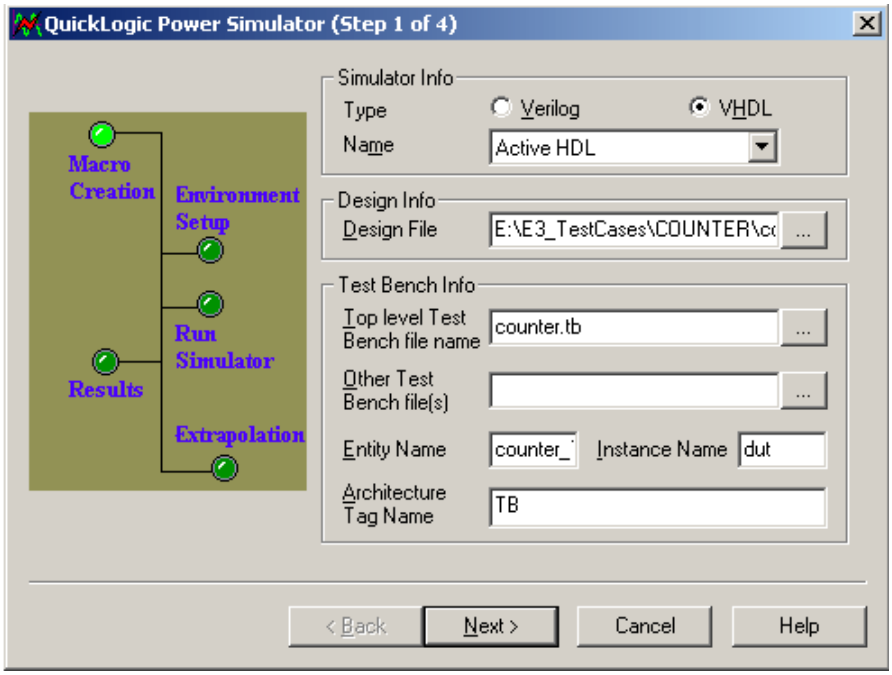
9.3.2 Step 1: Creating Simulation Macro and Simulation Task file

For Verilog flow:



© 2008 QuickLogic Corporation

For VHDL flow:



1. **Simulator Info**—Select the type of simulation by clicking on **Verilog** or **VHDL** radio buttons, and select the simulator to be used from the Name list.
2. **Design Info**—Enter the back-annotated design file (`.chp`) you want to use, including the path, or click the browse button ... to select the file.
3. **Test Bench Info**—Enter the following information:
   - **Top level Test Bench file name** without the path where the design is to be simulated, or click the browse button ... to select the file.
   - **Other Test Bench file(s)** written for the design to be simulated, if there are any, or click the browse button ... to select the file.
   - For Verilog, top level **Module Name** (required only for the ModelSim simulator), and for VHDL, **Entity Name**.
   - Top level **Instance Name**.
   - For VHDL, **Architecture Tag Name**

© 2008 QuickLogic Corporation

4. After making all the required selections, click **Next**.
The Power Simulator application generates the simulator macro file (`.do`) and the simulator task file (`.mtf/.mtb`), which is included in the top-level test bench file. **Table 9-1** lists the files generated by the Power Simulator application in Step 1.

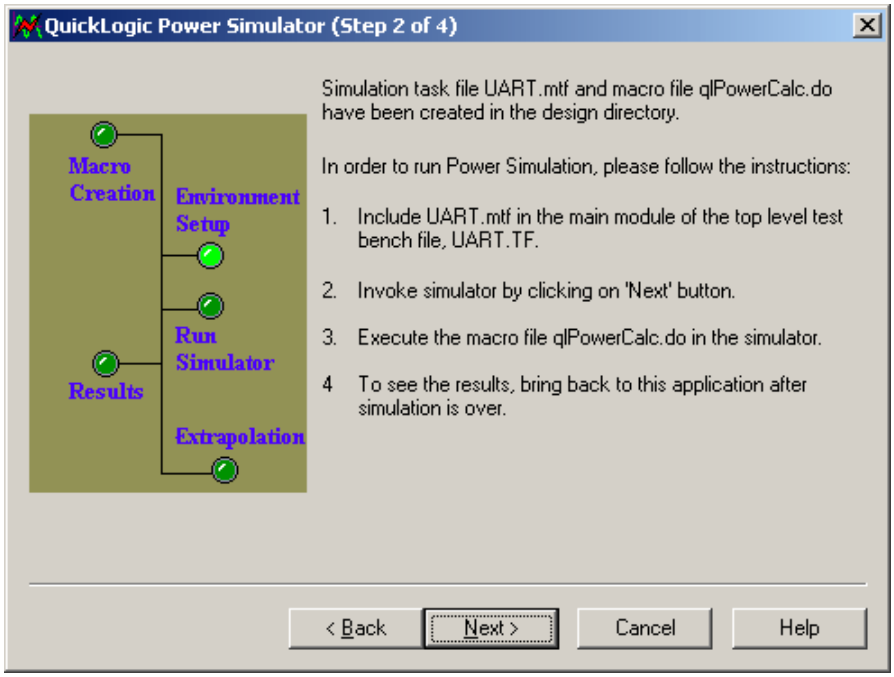
Table 9-1: Generated Files (Step 1)

File Name	Description
`qlPowerCalc.do`	The simulation macro file (`.do`) is created by clicking the **Next** button in Step 1. This generated macro file can be executed directly from the tools menu of the Active-HDL simulator or through the `Do qlPowerCalc.do` command in the ModelSim simulator.
`.mtf` (for Verilog only)	The simulation task file created in Step 1 should be included in the top level test bench of the selected design.
`.mtb` (for VHDL only)	The content of the simulation task file created in Step 1 should be included in the top level test bench.

9.3.3 Step 2: Setting Up Simulation Environment and Activating Simulator

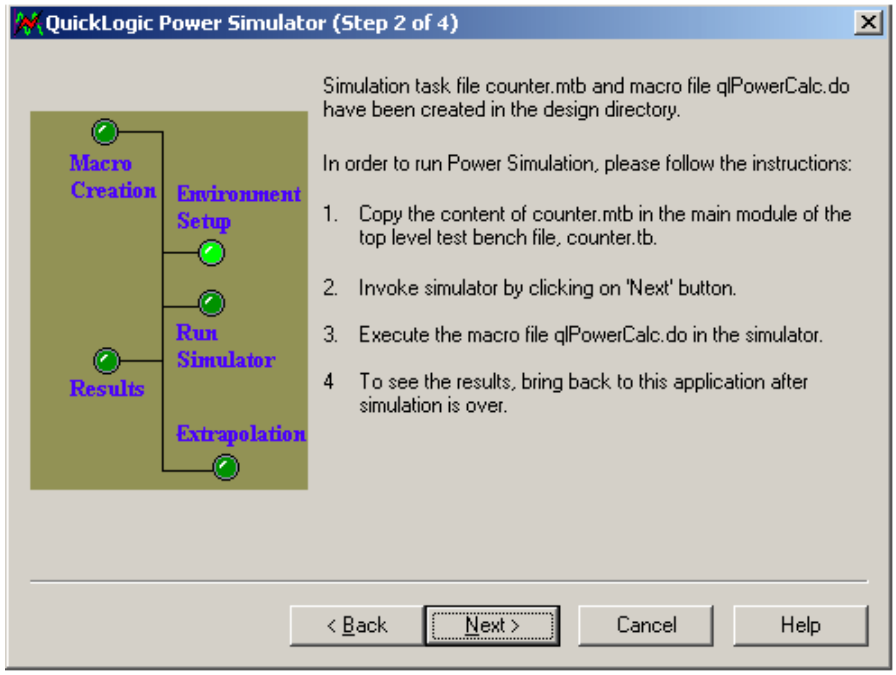
The dialog box in Step 2 explains the steps involved in the simulation environment setup process for the design to be simulated through the selected simulator.

For Verilog:


QuickLogic Power Simulator (Step 2 of 4)
Simulation task file UART.mtf and macro file qlPowerCalc.do have been created in the design directory.
In order to run Power Simulation, please follow the instructions:
1. Include UART.mtf in the main module of the top level test bench file, UART.TF.
2. Invoke simulator by clicking on 'Next' button.
3. Execute the macro file qlPowerCalc.do in the simulator.
4 To see the results, bring back to this application after simulation is over.
Macro Creation
Environment Setup
Run Simulator
Results
Extrapolation
< Back
Next >
Cancel
Help

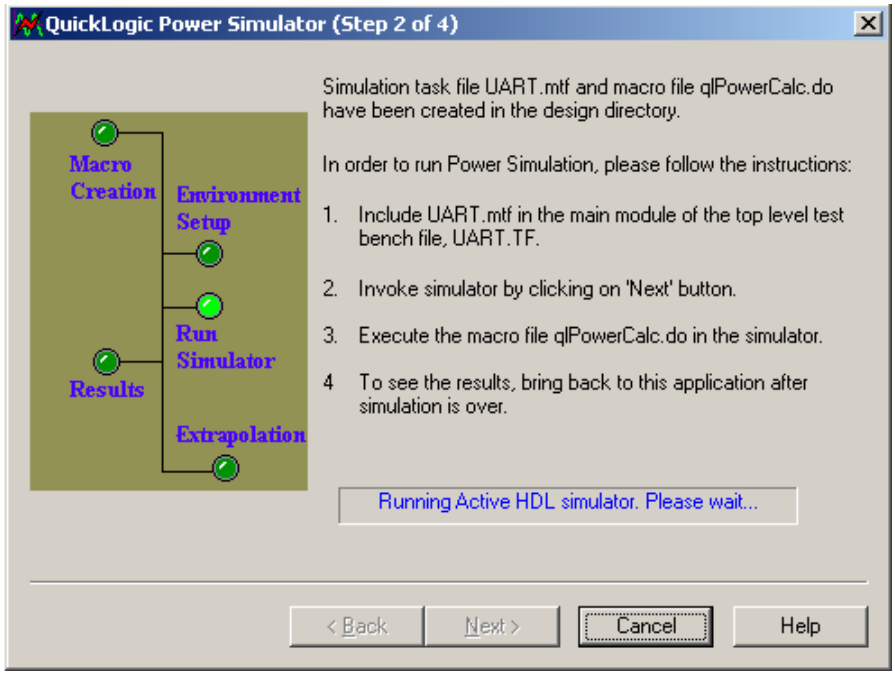
© 2008 QuickLogic Corporation

For VHDL:

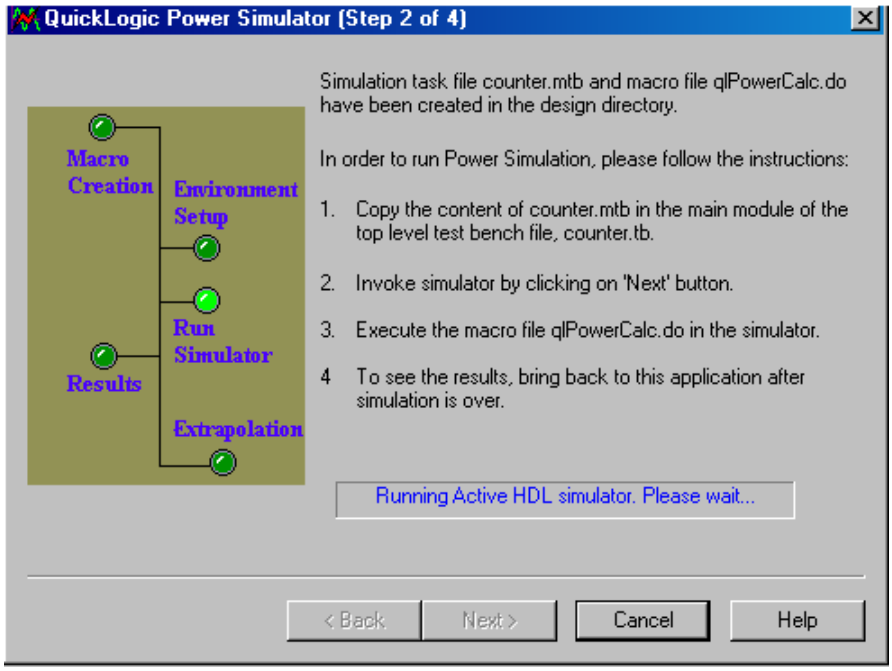

QuickLogic Power Simulator (Step 2 of 4)
Macro Creation
Environment Setup
Run Simulator
Results
Extrapolation
Simulation task file counter.mtb and macro file qlPowerCalc.do have been created in the design directory.
In order to run Power Simulation, please follow the instructions:
1. Copy the content of counter.mtb in the main module of the top level test bench file, counter.tb.
2. Invoke simulator by clicking on 'Next' button.
3. Execute the macro file qlPowerCalc.do in the simulator.
4 To see the results, bring back to this application after simulation is over.
< Back
Next >
Cancel
Help

After setting up the environment as described in the Step 2, click **Next**. The simulator is invoked and a status message is displayed at the bottom of the dialog box. (The **Next** button becomes disabled until the simulation completes.)

For Verilog flow:

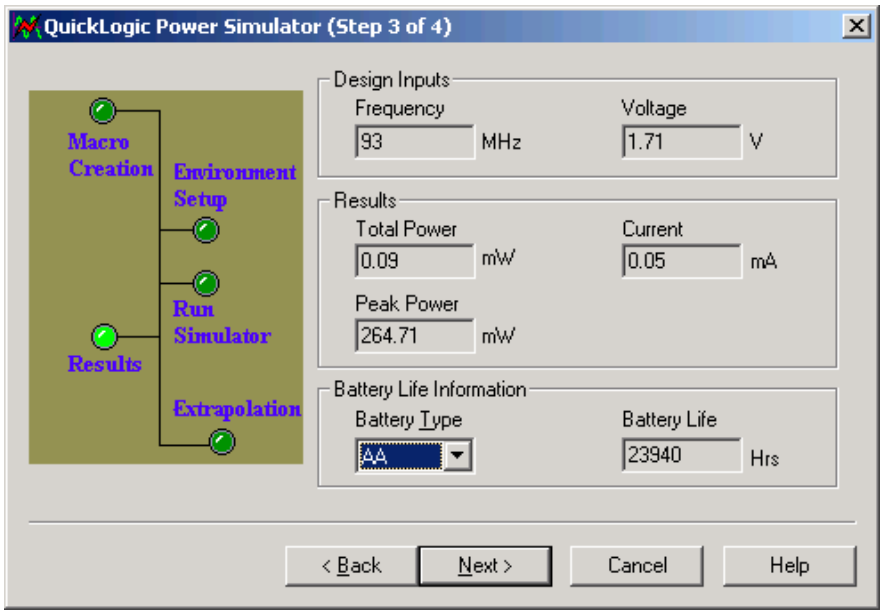

QuickLogic Power Simulator (Step 2 of 4)
Macro Creation
Environment Setup
Run Simulator
Results
Extrapolation
Simulation task file UART.mtf and macro file qlPowerCalc.do have been created in the design directory.
In order to run Power Simulation, please follow the instructions:
1. Include UART.mtf in the main module of the top level test bench file, UART.TF.
2. Invoke simulator by clicking on 'Next' button.
3. Execute the macro file qlPowerCalc.do in the simulator.
4 To see the results, bring back to this application after simulation is over.
Running Active HDL simulator. Please wait...
< Back
Next >
Cancel
Help

For VHDL flow:


QuickLogic Power Simulator (Step 2 of 4)
Simulation task file counter.mtb and macro file qlPowerCalc.do have been created in the design directory.
In order to run Power Simulation, please follow the instructions:
1. Copy the content of counter.mtb in the main module of the top level test bench file, counter.tb.
2. Invoke simulator by clicking on 'Next' button.
3. Execute the macro file qlPowerCalc.do in the simulator.
4 To see the results, bring back to this application after simulation is over.
Macro Creation
Environment Setup
Run Simulator
Results
Extrapolation
Running Active HDL simulator. Please wait...
< Back
Next >
Cancel
Help

9.3.4 Step 3: Generating Power Simulator Output

In Step 3, the Power Simulator results are generated automatically after the successful completion of the design simulation.



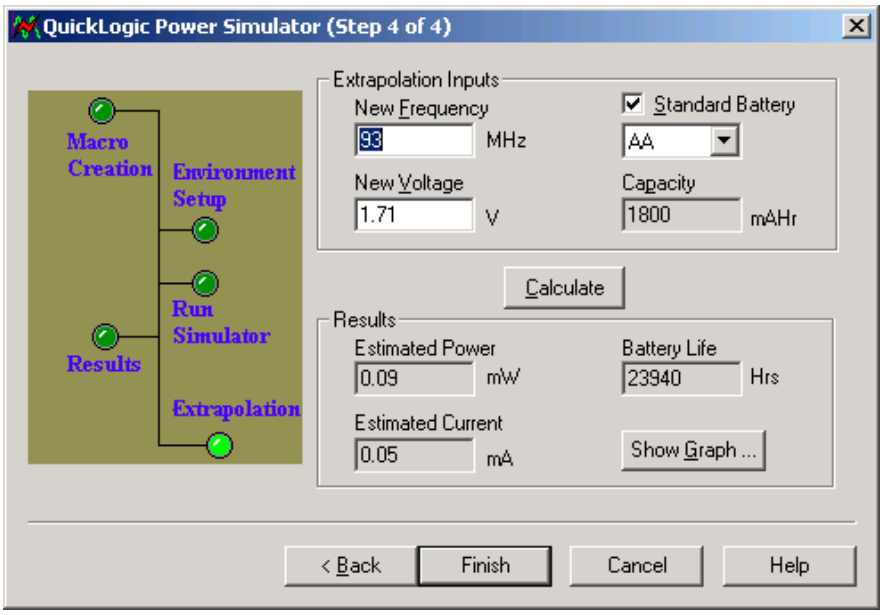
1. **Design Inputs**—The design's operating frequency and operating voltage are reported in this section.

© 2008 QuickLogic Corporation

2. **Results**—The calculated total dynamic power and peak power utilization of the design during its defined simulation period are displayed in this section.
3. **Battery Life Information**—The battery life of the device for the given design is calculated with respect to the calculated total power of the design. Battery life, measured in hours, can be viewed for the batteries listed in the Battery Type box.
4. Clicking **Next** initiates the Power Extrapolation tool.

9.3.5 Step 4: Using Power Extrapolation Tool

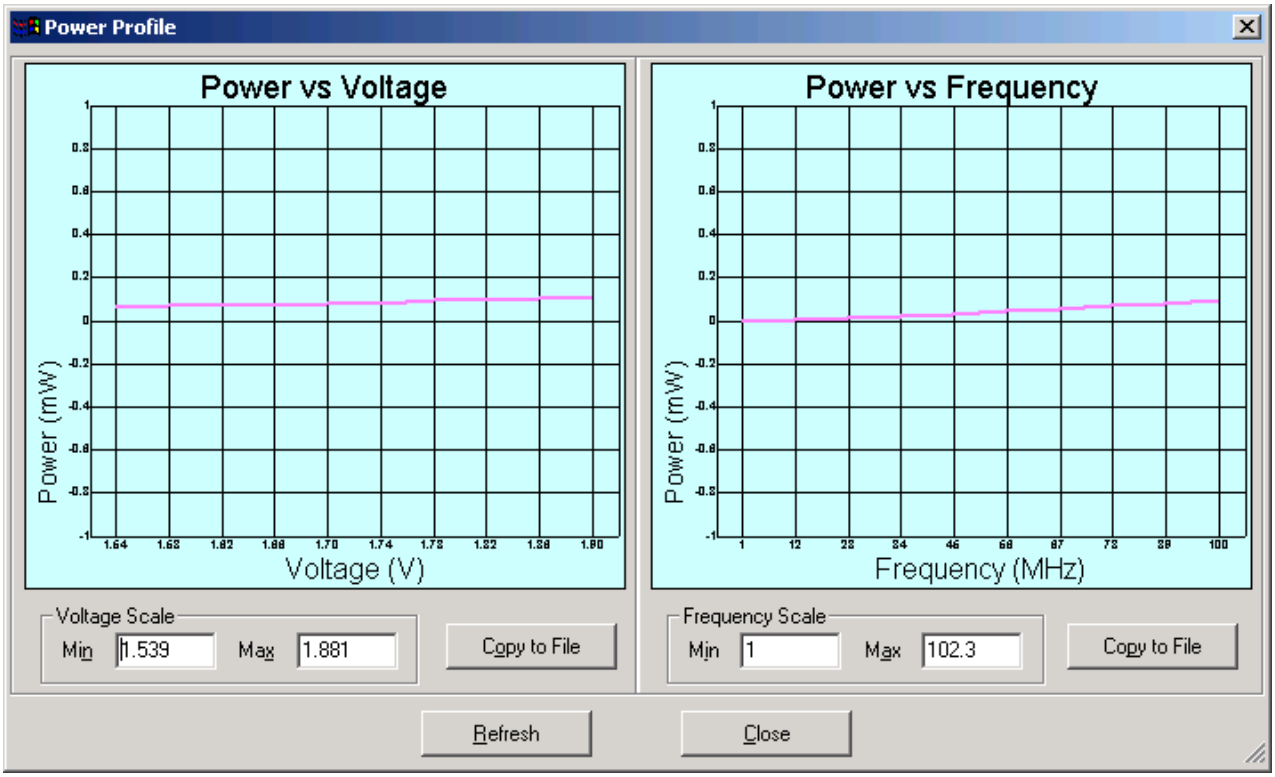
The Power Extrapolation tool estimates the total power consumption of the design for varying operating voltage and operating frequency:



1. **Extrapolation Inputs**—The design's operating frequency and voltage are shown as its inputs by default. The Power Extrapolation tool also accepts the standard battery types (AA or AAA). If the **Standard Battery** check box is selected, you can choose the standard battery type from the combo box. If the check box is cleared, you have to enter the capacity of the non-standard battery.
2. **Calculate**—Click Calculate for a given operating frequency, operating voltage and battery type to determine the estimated power and battery life.
3. **Results**—The estimated power and battery life are reported as output results for a given set of inputs in this section.
   Click **Show Graph** to display the power extrapolation graphs (*Power vs. Frequency* or *Power vs. Voltage*).

© 2008 QuickLogic Corporation

4. In the Power Profile dialog box, after varying frequency and voltage in the Extrapolation Inputs section, you can view the power extrapolation graphs.



To view the power profile for a given range of voltage/frequency, enter the min/max values for voltage/frequency scales and click **Refresh**.

9.4 Limitations

The new version of the Power Simulator application supports Active-HDL and ModelSim simulators for Verilog HDL flow and Active-HDL simulator support for VHDL flow. The power consumption reporting is performed for the entire design, not on functional block (IP block) basis, for any given back-annotated design in SpDE.

© 2008 QuickLogic Corporation

Chapter 10
Retarget Devices

The *retarget* feature in SpDE provides the user an option to change the target device during the loading a design netlist to SpDE for Place-and-Route. This eliminates the necessity of re-running the synthesis tool just to change the target device. Moreover, it also provides a means to select new devices that are not listed in the QuickLogic device family in the third party synthesis tools.

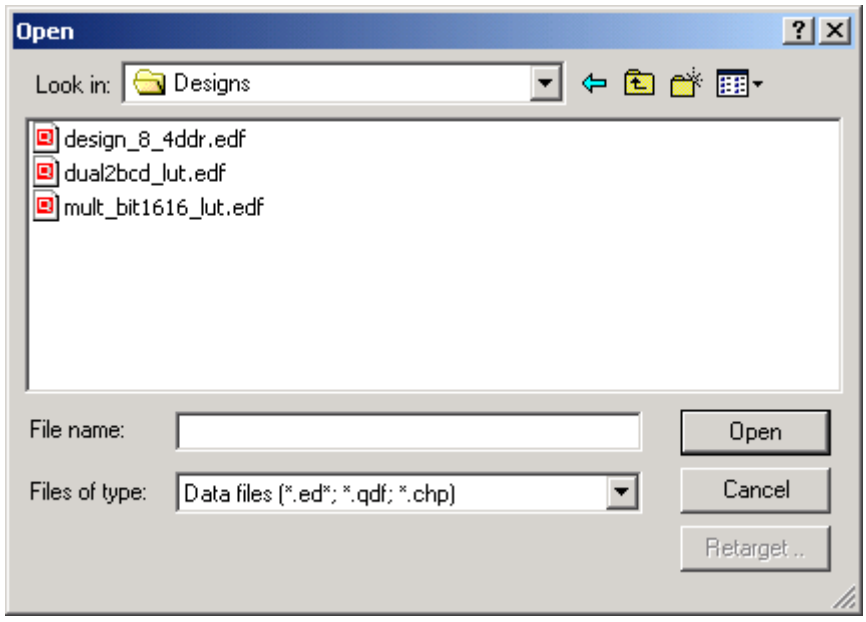
This chapter contains the following sections:

- **"Using the Retarget Devices Function" on page 119**

10.1 Using the Retarget Devices Function

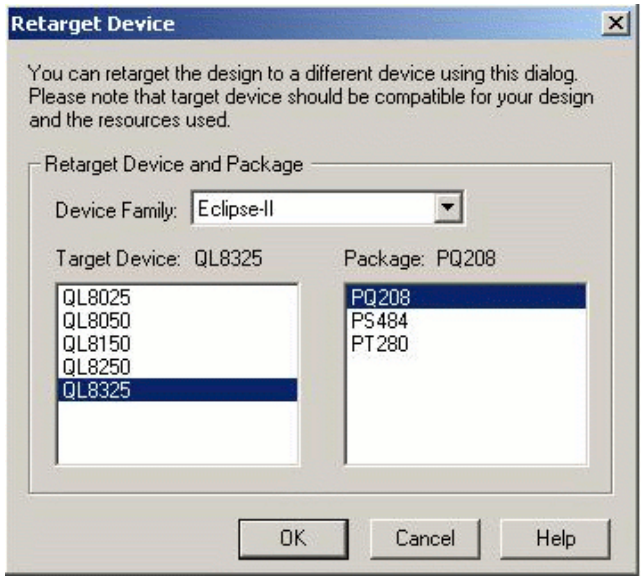
The retargeting of a part/package is done using the **Retarget** button and the Retarget Device dialog box.

1. Select **File>Open**, or click the **Open** icon, to open the Open dialog box.



© 2008 QuickLogic Corporation

2. Select the `.qdf/.edf` file to import and click the **Retarget..** button. A Retarget Device dialog box opens.



The Retarget Device dialog box contains the device family combo box, a Target Device (part) list box, and a Package list box. The items in the Target Device list box get updated based on the selection in the Device Family combo box. The items in the Package list box get updated based on the selection in the Target Device list box.

The retarget device comes with default part and package selections. These default selections read from one of the following:

- The `.qcf` file.
- If the `.qcf` file does not exist or the part/package name is not mentioned in it, the part/package name is read from the design file (`.qdf /.edf`).
- If none of the files above are available, the part/package name is read from the defaults specified in the **Load>Save** tab of the **Tools>Options** dialog box.

3. Click the **OK** button to generate/update the selected part/package name in the corresponding `.qcf` file, or click the **Cancel** button to return to the Open dialog box.

If you intend to import Verilog, VHDL, or Precision RTL projects through the **File>Import Using Precision RTL** menu, the device can be retargeted through the Retarget Device dialog box that appears before the `.qdf/.edf` file is located in SpDE.

The format used to specify the part/package name in the `.qcf` file is as follows:

```
#Part Name
partname <part name>
#Pack Name
packname <package name>
```


© 2008 QuickLogic Corporation

Chapter 11
Design Techniques

This chapter discusses several techniques for optimizing designs and contains the following sections:

- **"Using Automatic Buffering Tools" on page 121**
- **"Evaluating the Results of Auto-Buffering" on page 123**
- **"Inserting Schematic Buffers to Speed up the Design" on page 125**
- **"Inserting Buffers in VHDL and Verilog Designs" on page 128**
- **"Pipelining to Increase Clock Frequency" on page 130**
- **"Verilog State Machine Design" on page 130**

11.1 Using Automatic Buffering Tools

QuickWorks offers two tools that can perform automatic buffering of high fan-out nets:

- Precision RTL Synthesis
- SpDE

11.1.1 How Automatic Buffering Works

Automatic buffering often speeds up a design that has been slowed down due to high fan-out nets. However, it is best to understand how the automatic buffering algorithm works in case the automatic buffering results are not optimal for your design.

11.1.1.1 Determining Critical Paths

Automatic buffering looks at the entire design and determines the critical paths by looking at the number of logic cells between flip-flops and I/O pins. The paths that contain the largest number of logic cells between these end points are considered most critical. The paths with the smallest number of logic cells between these end points are considered least critical.

11.1.1.2 Choosing Nets to Buffer

High fan-out nets in critical paths (see the previous section) are first to receive buffering treatment. The logic optimizer adds buffers in a buffer tree format, putting fewer buffers in paths that are most critical, and putting more buffers in the paths that are less critical. This is great for synchronous designs that operate on one clock, but may not be optimal for all designs. The Auto-buffering tool also looks at the placement of the logic cells to determine which points on a net should be buffered for best results.

Try auto-buffering before manual buffering because it is easier. If the results are not adequate, then try manual buffering. See "Evaluating the Results of Auto-Buffering" on page 123.

© 2008 QuickLogic Corporation

11.1.2 Using the SpDE Auto-Buffering Algorithm

When you place and route a design, one of the tools run in SpDE is the Auto-buffering tool. In **SpDE**, select **Tools>Options** to open a dialog box used to change tool settings.

To turn on SpDE's auto-buffering algorithm (which will automatically buffer high fan-out nets in the critical paths of the design), click on the **Logic Optimizer** tab and select the check box titled **Auto Buffer Insertion**. Click **Save Setting** to record this change for all future SpDE sessions in the `SPDE.INI` file.

For more information, see also:

- Constraint File Format (.QCF File) [PDF]

11.1.3 Disabling SpDE Auto-Buffering Net-by-Net

SpDE has a special feature that allows designers to disable the Auto-buffering algorithm on a net-by-net basis. You might want to do this when you have a high fan-out net that is not speed-critical and you want to free up logic cells so that other critical nets can be buffered.

Edit or create a `.qcf` (QuickLogic Constraint File), and add the following line:

```
dont buffer <net_name1> <net_name2> <net_name3>...
```

This line will be read into SpDE along with the design when you select **File>Import**. With this command in the `.qcf` file, each net name listed after: `dont buffer` will be ignored by the Auto-buffering algorithm, freeing up logic cell resources for other nets to be buffered.

For more information, see also:

- Constraint File Format (.QCF File) [PDF]

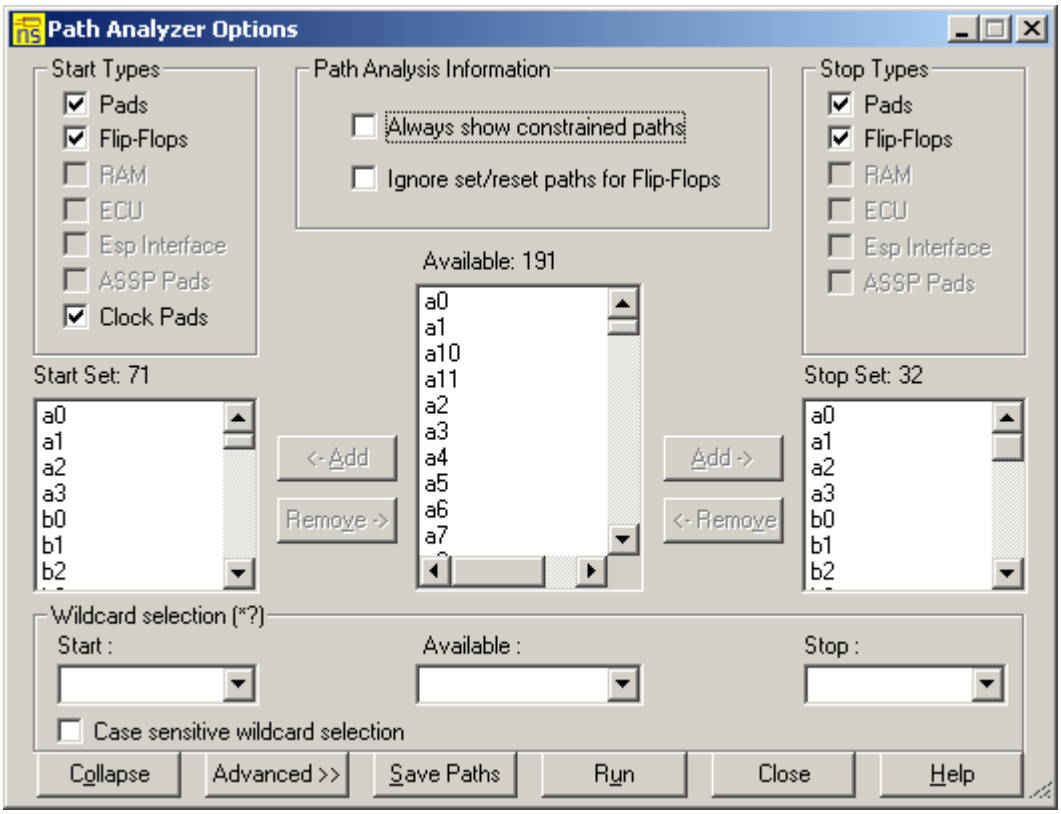
© 2008 QuickLogic Corporation

11.2 Evaluating the Results of Auto-Buffering

Automatic buffering can be done by Precision RTL (during HDL synthesis) or by SpDE. The best performance is usually achieved by enabling only one of these auto-buffering tools—not both. See the previous section to determine how to enable or disable these buffering tools.

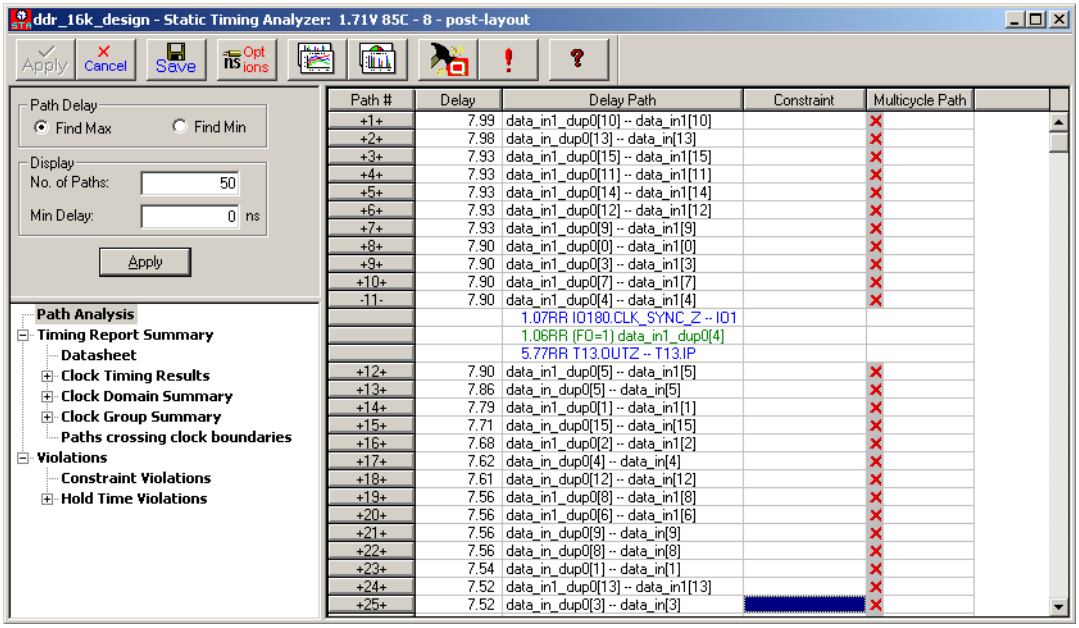
11.2.1 Finding Auto-Buffered Nets in the Path Analyzer

After running all tools on the design (including any auto-buffering tools), the Path Analyzer can determine the performance of the design. Due to the way nets are named by the Auto-buffering tools, the Path Analyzer can also see the effect of the Auto-buffering algorithms. Open the Path Analyzer by using the STA toolbar button or by selecting **Tools> Static Timing Analyzer>Path Analysis** from the SpDE menu bar. The Path Analyzer Options dialog box appears.



If your critical path is not the first path in the Path Analyzer, you may need to select a different Start and Stop set of nets.

Once you have located a critical path, you can examine the path in more detail by expanding the path. Double-click on the path number in the **Path #** column of the Path Analyzer. After expanding the critical path, the Path Analyzer dialog box will contain the following information.


ddr_16k_design - Static Timing Analyzer: 1.71V 85C - 8 - post-layout
Apply
Cancel
Save
Options
Path Delay
Find Max
Find Min
Display
No. of Paths: 50
Min Delay: 0 ns
Apply
Path Analysis
Timing Report Summary
Datasheet
Clock Timing Results
Clock Domain Summary
Clock Group Summary
Paths crossing clock boundaries
Violations
Constraint Violations
Hold Time Violations
Path # | Delay | Delay Path | Constraint | Multicycle Path
+1+ 7.99 data_in1_dup0[10] -- data_in1[10]
+2+ 7.98 data_in_dup0[13] -- data_in[13]
+3+ 7.93 data_in1_dup0[15] -- data_in1[15]
+4+ 7.93 data_in1_dup0[11] -- data_in1[11]
+5+ 7.93 data_in1_dup0[14] -- data_in1[14]
+6+ 7.93 data_in1_dup0[12] -- data_in1[12]
+7+ 7.93 data_in1_dup0[9] -- data_in1[9]
+8+ 7.90 data_in1_dup0[0] -- data_in1[0]
+9+ 7.90 data_in1_dup0[3] -- data_in1[3]
+10+ 7.90 data_in1_dup0[7] -- data_in1[7]
-11- 7.90 data_in1_dup0[4] -- data_in1[4]
1.07RR IO180.CLK_SYNC_Z -- IO1
1.06RR (FO=1) data_in1_dup0[4]
5.77RR T13.OUTZ -- T13.IP
+12+ 7.90 data_in1_dup0[5] -- data_in1[5]
+13+ 7.86 data_in_dup0[5] -- data_in[5]
+14+ 7.79 data_in1_dup0[1] -- data_in1[1]
+15+ 7.71 data_in_dup0[15] -- data_in[15]
+16+ 7.68 data_in1_dup0[2] -- data_in1[2]
+17+ 7.62 data_in_dup0[4] -- data_in[4]
+18+ 7.61 data_in_dup0[12] -- data_in[12]
+19+ 7.56 data_in1_dup0[8] -- data_in1[8]
+20+ 7.56 data_in1_dup0[6] -- data_in1[6]
+21+ 7.56 data_in_dup0[9] -- data_in[9]
+22+ 7.56 data_in_dup0[8] -- data_in[8]
+23+ 7.54 data_in_dup0[1] -- data_in[1]
+24+ 7.52 data_in1_dup0[13] -- data_in1[13]
+25+ 7.52 data_in_dup0[3] -- data_in[3]

NOTE: In the Path Analyzer dialog box displaying the expanded path, the column width of the Delay Path column has been increased so that the entire net-to-net path name of each path can be viewed.

Any net that has a string such as `.I4-N_3` in its name has been inserted on the output of a new buffer created by SpDE's auto-buffering tools.

Any nets with strings such as `buf0` or `buf1` have been inserted on the outputs of buffers created by Precision's auto-buffering tools. You should not see instances of both of these net name types. If you do, disable one of the auto-buffering tools – see **"Disabling SpDE Auto-Buffering Net-by-Net" on page 122** for more information.

Also, at the beginning of each sub-path, you will see a string such as (FO=2). This refers to the fan-out of the start net of the path.

With this information you should now be able to use the Path Analyzer to determine if buffers have been automatically added to the critical paths of your design, and if so, what effect they have had on the fan-outs and the delay. If the results are not satisfactory, then you may be able to improve them by inserting manual buffers in your design.

For more information, see also:

- **"Path Analysis" on page 322** - Overview
- **"Path Analyzer Options" on page 323** - Using Path Analyzer Tool

© 2008 QuickLogic Corporation

11.3 Inserting Schematic Buffers to Speed up the Design

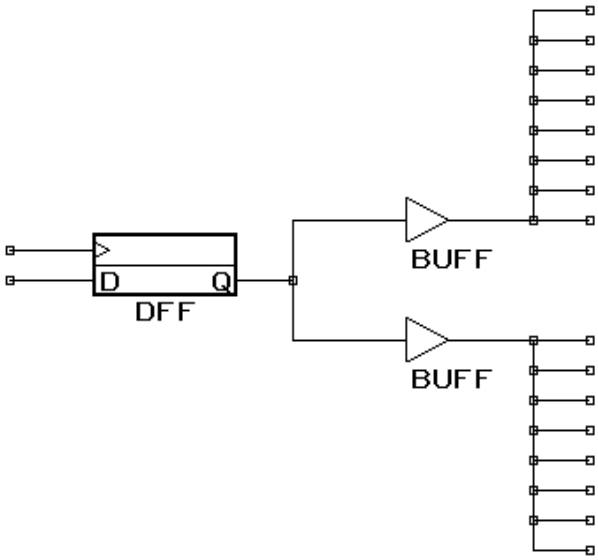
For high fan-out, timing-critical nets, designers should consider improving design performance using buffering techniques. In some cases, solutions such as paralleling or pipelining can be used. This section describes four techniques that can be used to improve circuit performance:

- Split Buffering
- Selective Buffering
- Paralleling
- Double Buffering

11.3.1 Split Buffering

As demonstrated in the circuit, as shown in **Figure 11-1**, split buffering breaks a wide-fan-out net into two or more nets. Without the buffers, the `DFF` macro drives a fan-out of 16. As shown, the `DFF` macro drives a fan-out of two, and each `BUFF` macro drives a fan-out of eight.

Figure 11-1: Circuit Demonstrating Split Buffering



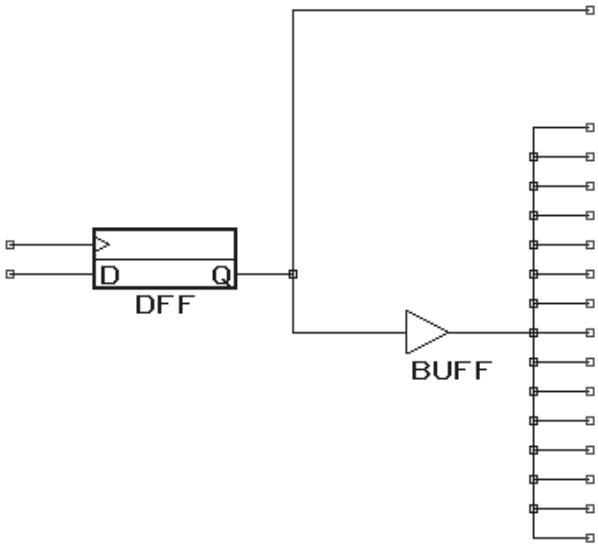
NOTE: Adding buffers introduces a logic cell delay to the net. This added delay must be balanced against the gain in reducing the fan-out. Simple split buffering, as demonstrated in **Figure 11-1**, is generally employed only with fan-outs ≥ to eight.

11.3.2 Selective Buffering

Selective buffering can be employed very effectively in situations where a high fan-out net has a small number of critical destinations and a large number of non-critical destinations. Suppose the `DFF` macro drives a fan-out of 16, but only 1 of the destinations is in the critical path of the circuit. As demonstrated in **Figure 11-2 on page 126**, a single buffer is inserted for the 15 non-critical destinations. The `DFF` macro drives a fan-out of two, as in the earlier example, but now there is no added logic cell delay in the critical path.

© 2008 QuickLogic Corporation

Figure 11-2: Circuit Demonstrating Selective Buffering

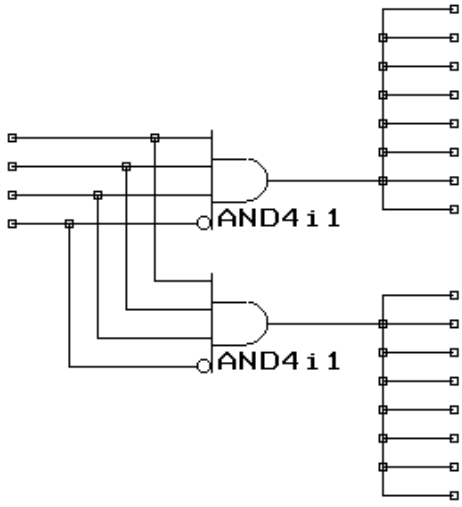


NOTE: As mentioned earlier, buffers should be introduced with care and skill. Selective buffering offers tremendous improvement in circumstances where the circuit has a few clearly identifiable critical paths.

11.3.3 Paralleling

As illustrated in the example in **Figure 11-3**, successful buffering must balance reduced fan-out against the added delay through the buffer. Paralleling is an alternative that does not introduce this added delay. Paralleling duplicates the logic driving a high fan-out load to reduce the effective fan-out.

Figure 11-3: Circuit Demonstrating Paralleling



© 2008 QuickLogic Corporation

Figure 11-3 shows the paralleling technique: the AND gate has been duplicated, with each of its inputs tied to the corresponding input on the twin gate. Each AND gate drives a fan-out of eight, effectively halving the fan-out without introducing the added delay associated with buffering. By duplicating the AND gate, however, the fan-out on each of the input nets is increased.

Notice that Paralleling is similar to Double Buffering (see **"Double-Buffering" on page 127**), except that the outputs are not tied together. Paralleling should be used instead of Double Buffering when:

- The skew is not critical
- Too many express wires have already been used for high-drive inputs or double-buffers (see **Chapter 14, "Router" on page 155**)
- The logic to be replicated will not fit into an AND fragment of the logic cell (no larger than an `AND6i3`)

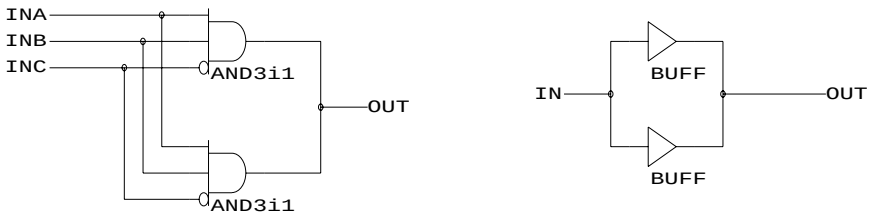
For more information, see also:

- **Chapter 14, "Router" on page 155** - Routing Resources

11.3.4 Double-Buffering

Certain architectures (pASIC3, QuickRAM, Eclipse, and EclipsePlus) allow a net to be driven by two sources in specific cases. This is called *double-buffering*. Using two gates to drive a high fan-out net speeds up the performance of the net dramatically. Examples of using double-buffering in schematics appear in **Figure 11-4**.

Figure 11-4: Double-Buffering Examples



Double-buffering is legal as long as the two gates driving the high fan-out net are identical gates, with the same nets on the inputs and the output. Each gate must fit into an AND-Fragment, so the largest available gate is the `AND6i3` (6-input AND gates with three inverted inputs). Nets driven by double buffers are routed to the top or bottom channel of the device before being routed to the loads, so this solution should only be used in cases where the placement of the loads is well defined and low skew is the primary motivation. If the primary motivation is to reduce delay, not skew, then split buffering produces much better results.

For more information, see also:

- **Chapter 14, "Router" on page 155** - Routing Resources

© 2008 QuickLogic Corporation

11.4 Inserting Buffers in VHDL and Verilog Designs

When optimizing for speed in VHDL and Verilog designs, it is sometimes necessary to manually insert buffers into the design. Before inserting buffers in synthesis, you should first turn off the Synthesis Tool Buffering (see the section **"Using Automatic Buffering Tools" on page 121**). If automatic buffering does not meet the needs of the design, consider manually buffering high fan-out nets.

The techniques used for schematic buffering (described in the previous section) can also be applied to VHDL and Verilog designs. However, just inserting a behavioral buffer (i.e. **`buffa = a`**) in the HDL file results in this buffer being optimized out of the design. Therefore, you have to instantiate a special QuickLogic buffer into the VHDL or Verilog design that will not be optimized away.

For more information, see also:

- **"Using Automatic Buffering Tools" on page 121**

11.4.1 Instantiate a Special Buffer in Verilog

1. Add an 'include line at the top of your Verilog design file to include the QuickLogic macros file: `'include "c:\pasic\spde\data\macros.v"`, assuming `c:\pasic\` is the directory in which QuickLogic QuickWorks is installed in. Add the buffer because you need it in your design. For example, to split-buffer a signal called `dramq` into three buffers, you would add the following lines in your Verilog module:

   Syntax:

   ```
   buffer_name instance_name (input_signal,output_signal);
   ```

   Example:

   ```
   buff qlbuff1 (dramq, dramqa);
   buff qlbuff2 (dramq, dramqb);
   buff qlbuff3 (dramq, dramqc);
   ```

2. Split up the loads for `dramq` into three parts, each driven by `dramqa, dramqb,` and `dramqc` (respectively).
3. To prevent the Synthesis tool from removing these buffers, you must put a `SYN_MACRO` attribute on any module that contains the buffers. This is accomplished with the following parameter command at the top of the Verilog module you will insert the buffers in:

   ```
   parameter syn_macro = 1; /* syn_macro must be lower  case*/
   ```

11.4.2 Instantiate a Special Buffer in VHDL

VHDL does not have an **`'include`** command to automatically include a different file.

1. Specify the file: C:\pasic\spde\data\macros.vhd, assuming c:\pasic\ is the directory in which QuickLogic QuickWorks is installed in, to the Precision RTL Synthesis tool before compilation.
2. In the line before the entity declaration in which you need to use the buffer, tell the VHDL compiler to use the `ql_macros` package:

   ```
   use work.ql_macros.all;
   ```


© 2008 QuickLogic Corporation

Add this line before every entity declaration in which you need to use the buffer.

3. Inside the architecture, instantiate the buffer with standard VHDL instantiation. For example, to split-buffer a signal called `dramq` into three buffers, you would add the following lines in your VHDL module:

   Syntax:

   ```
   instance_name: buffer_name PORT MAP (input_signal, output_signal);
   ```

   Example:

   ```
   qlbuff1: buff port map (dramq, dramqa);
   qlbuff2: buff port map (dramq, dramqb);
   qlbuff3: buff port map (dramq, dramqc);
   ```

   Split up the loads for `dramq` into three parts, each driven by `dramqa`, `dramqb`, and `dramqc`, respectively.

4. To prevent the synthesis tool from removing these buffers, you must put a `SYN_MACRO` attribute on any module that contains the buffers. This is accomplished with the following attribute command at the top of the VHDL architecture in which you insert the buffers (replace <architecture name> with the name of the architecture in which you are inserting these attributes:

   ```
   attribute syn_macro: integer;
   attribute syn_macro of <architecture name>: architecture is 1;
   ```

© 2008 QuickLogic Corporation

11.5 Pipelining to Increase Clock Frequency

Pipelining is a technique widely used to increase system clock rate. By inserting registers in long combinatorial paths, the length of the critical path can be shortened and operations can be overlapped, increasing the system clock rate. The pASIC architecture promotes pipelining, as each logic cell contains a D flip-flop. As a result, a design can be pipelined with little or no increase in the number of logic cells used.

For more information, see also:

- **"Fan-out Reduction Techniques" on page 294**
- **"Reducing Set-up Time and Clock-to-Out" on page 294**

11.6 Verilog State Machine Design

State machines can be implemented in a number of different ways. Two of the most popular implementations are encoded state and one-hot encoding respectively.

Encoded state machines require fewer flip-flops than one-hot state machines, but may require more combinatorial logic to generate flip-flop inputs.

One-hot state machines, on the other hand, require more flip-flops, but typically less state decode logic. Due to this fact, and because of the register-rich nature of these devices, they are especially well-suited to be implemented in FPGAs. These conditions also enable most one-hot state machines to run at higher speeds than their encoded counterparts. Because QuickLogic logic cells have a high number of inputs, dense, encoded state machines can sometimes produce an optimal result.

Although a state machine can be entered in the form of schematics or Boolean equations (QuickBoolean), Verilog makes state machine description especially easy. An HDL description of a state machine can be entered very quickly, with most of the work (logic generation) done by the synthesis tool. With the help of a Verilog simulator, such as Active HDL, simulation and verification of a state machine's operation can also be done quickly and efficiently.

This section describes the two most popular methods of implementing state machines in Verilog: *encoded state* and *one-hot*.

11.6.1 Encoded State Machines

Verilog state machines are best implemented with case statements. A series of `if` statements or nested `if` statements often results in an improper or less-than-optimal result.

Before writing the case statement, it is often useful to define the values for each state with parameters. Parameters make Verilog code more readable, and allow changes, such as the re-assignment of state bits, to be made more easily. The syntax for defining parameters in Verilog is as follows:

```
parameter <parameter_name> = <value>;
```

Example: `parameter state_refresh = 4'b0010;`

© 2008 QuickLogic Corporation

Please note that a parameter definition must be made within the module that uses it, not outside the module or in another module. Also, `define` statements can be used instead of parameters, but are considered bad coding style if they're not necessary.

Once parameters have been defined for each state, you can write the case statement that describes the state machine behavior. A case statement behaves similarly to a series of nested `if` statements, where each `if` statement compares the current state against a fixed (known) value. The advantage of using a case statement is that it's easier to understand and results in the most optimal implementation when the design is synthesized. On the other hand, nested `if` statements may not produce an optimal result because this type of `if-else` structure implies priority. The syntax for a Verilog case statement is shown below:

```
case (variable)
value1:<statement>
value2:<statement>
...
endcase
```

When the case statement is executed the current value of variable is compared with the branches of the case statement. When a match is found, the statement associated with that branch is executed and the case statement is exited. If multiple statements need to be executed for any one branch, a begin…end block can be used.

© 2008 QuickLogic Corporation

Outlined below is an example of a simple state machine, including its implementation in Verilog.

```
State Definitions
S0 00
S1 01
S2 10
S3 11
module encoded (clk, rst, in, state);
input clk, rst;
input [3:0] in;
output [1:0] state;
reg [1:0] state;
parameter s0 = 2'b00;
parameter s1 = 2'b01;
parameter s2 = 2'b10;
parameter s3 = 2'b11;
parameter sx = 2'bx; // don't-care state, used for default case
always @(posedge clk or posedge rst)
if (rst)state <= s0;
else case  (state)
s0:if(in == 4'h5)state <= s1;
else if(in == 4'h8)state <= s3;
else state <= s0;
s1:state <= s2;
s2:if(in == 4'hF)state <= s0;
else if(in == 4'hC)state <= s3;
elsestate <= s2;
s3:if(in == 4'h0)state <= s1;
elsestate <= s3;
default:beginstate <= sx;
$display ("warning: unknown state");
end
endcase
endmodule
```

As you can see, the use of the `case` statement in the above example helps improve the readability of the code. If nested statements are used, even for a simple design such as the one shown above, the code is difficult to read and may not produce an optimal implementation (the synthesis tool may try to build a priority encoder).

Please notice that the last case listed is called `default`. If the value of state doesn't match any of the branches, the `default` case is executed. The default case is typically included for completeness and to help verify the operation of the design. For example, when the simulation

© 2008 QuickLogic Corporation

begins, the state bits are unknown, because this is the behavior of the flip-flops in QuickLogic's pASICs. If the `CLK` signal triggers the `always` block before the `RST` signal clears the state bits to a known value, there isn't a case branch that equals the current value of the state bits. It is usually considered good practice to include a default case, because it can be used to flag possible design errors while simulating the design. In the above example, the $display simulation directive instructs the simulator to display the message: `warning: unknown state` if the state machine happens to enter an unspecified state. The $display directive is not required for the functionality of the state machine; it is simply included for simulation purposes.

11.6.2 One-Hot State Machines

The basic difference between an encoded state machine and a one-hot state machine lies in the encoding of the state bits. In a one-hot state machine, each state bit is used to represent only one state. Therefore, the number of state bits must be equal to the number of possible states in the state machine. One-hot state machines typically require less combinatorial logic feeding each flip-flop because the presence of any particular state can be found by examining the state bit corresponding to that state. By comparison, the detection of any one state in encoded state machines requires the decoding of all state bits, not just one.

The implementation of one-hot state machines in Verilog is almost identical to the implementation of encoded state machines. The only difference lies in the way the state bits are encoded. The reason each type of state machine may be implemented in similar ways lies in the quality of the synthesis tool included with QuickWorks. Precision RTL will automatically detect that a state machine has been one-hot encoded if you include a default case, as shown in the next example. The default case tells Precision RTL that all possible states have been specified in the case statement. If the states are one-hot encoded, additional logic will *not* be generated to decode all the state bits. Instead, the presence of any one state will be detected by only decoding the state bit assigned to it.

© 2008 QuickLogic Corporation

Shown below is the state machine from the previous example, using one-hot encoding. The following encoding scheme will be used:

```
State Definitions
S0          0001
S1          0010
S2          0100
S3          1000
```

```
module one_hot (clk, rst, in, state);
input clk, rst;
input [3:0] in;
output [3:0] state;
reg [3:0] state;

parameter s0 = 4'b0001,
          s1 = 4'b0010,
          s2 = 4'b0100,
          s3 = 4'b1000,
          sx = 4'bx;

always @(posedge clk or posedge rst)
          if (rst)                          state <= s0;
          else case (state)
                s0:    if     (in == 4'h5)  state <= s1;
                       else if(in == 4'h8)  state <= s3;
                       else          state <= s0;
                s1:                         state <= s2;
                s2:    if     (in == 4'hF)  state <= s0;
                       else if(in == 4'hC)  state <= s3;
                       else          state <= s2;
                s3:    if     (in == 4'h0)  state <= s1;
                       else                 state <= s3;
                // the following default case notifies Precision RTL
                // that all possible states have been specified above.
                default:     begin               state <= sx;
                       $display ("warning: unknown state");
                       end
          endcase
endmodule
```


© 2008 QuickLogic Corporation

Each of these designs were synthesized and placed and routed with QuickWorks. Some results are shown in **Table 11-1**.

Table 11-1: QuickWorks Results

	Logic Cells	FF's	Inputs	Outputs	Max. FF-FF Delay(-0 speed grade)
Encoded	6	2	6	2	13.7ns
One-Hot	7	4	6	4	8.8ns

The numbers shown in **Table 11-1** were obtained by placing and routing the two example designs. The timing analysis was done with a worst-case corner. As you can see, the one-hot state machine is slightly larger, due primarily to the fact that it required more flip-flops than the encoded state machine. The one-hot state machine, however, is capable of much higher speeds than the encoded state machine. This is often the case when implementing state machines in FPGAs; one-hot state machines frequently outperform encoded state machines, although they sometimes require more logic.

11.6.3 Alternative Coding Technique for One-Hot State Machines

While the coding method explained in the previous section is very easy to understand, it is possible that large complex state machines will not be implemented properly. For this reason we will present an alternative technique for writing one-hot state machines to improve your implementation results.

When designing large state machines, we recommend you use different coding techniques and choose the one that produces the best result. For example, instead of using only one set of parameters for both comparing and assigning state values, use two sets of parameters. One set can be used for comparing state bits and contains `don't-cares`. In this way, the specific bit indicating the presence (or absence) of a certain state can be isolated from the other state bits. The other set of parameters is used for assigning values to the state bits, and therefore contains fixed known values (`1`'s and `0`'s). Also, a `casex` statement is used instead of the normal `case`. This is because the state bit comparisons now contain *don't-care* bits. An example of this technique is shown below.

© 2008 QuickLogic Corporation

```
module one_hot_1 (clk, rst, in, state);
input clk, rst;
input [3:0] in;
output [3:0] state;
reg [3:0] state;

parameter s0 = 4'b0001,     t_s0: 4'bxxx1,
          s1 =   4'b0010,   t_s1: 4'bxx1x,
          s2 =   4'b0100,   t_s2: 4'bx1xx,
          s3 =   4'b1000,   t_s3: 4'b1xxx,
          sx =   4'bx;

always @(posedge clk or posedge rst)
          if (rst)                        state <= s0;
          else casex (state)
                t_s0:  if     (in == 4'h5)  state <= s1;
                       else if(in == 4'h8)  state <= s3;
                       else                 state <= s0;
                t_s1:                       state <= s2;
                t_s2:  if     (in == 4'hF)  state <= s0;
                       else if(in == 4'hC)  state <= s3;
                       else                 state <= s2;
                t_s3:  if     (in == 4'h0)  state <= s1;
                       else                 state <= s3;
                // the following default case notifies Precision RTL
                // that all possible states have been specified above.
                default:      begin         state <= sx;
                              $display ("warning: unknown state");
                              end
          endcase
endmodule
```

In most cases, both of the one-hot techniques will result in similar implementations. If this occurs, choose a technique based on your personal preference.

© 2008 QuickLogic Corporation

Chapter 12
Logic Optimizer

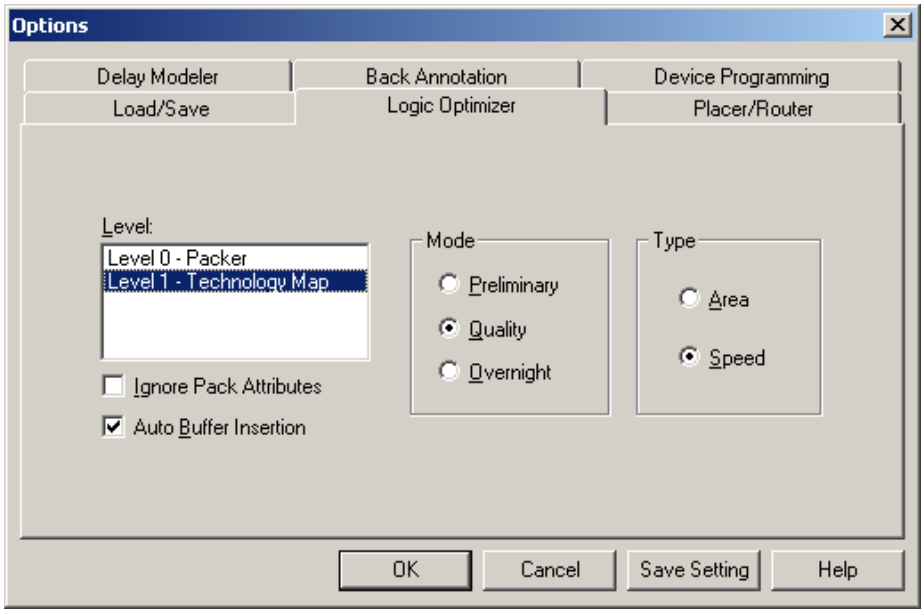
The QuickLogic Logic Optimizer is the first tool to be run after a design netlist has been loaded into SpDE. The Logic Optimizer uses sophisticated technology mapping algorithms to efficiently partition logic into Logic Cells. It contains the following sections:

- **"Partition Logic with the Logic Optimizer" on page 137**
- **"Choosing Area or Speed Optimization" on page 139**
- **"Logic Optimization Modes" on page 139**
- **"Using the Pack Attribute on Instances in a Design" on page 140**
- **"Automatic Buffer Insertion" on page 141**

12.1 Partition Logic with the Logic Optimizer

There are two levels of optimization available in SpDE: Level 0 and Level 1.

To select the optimization levels, select **Tools>Options** from **SpDE**. The Options dialog box opens.



12.1.1 Level 0: Packer

The Level 0 Packer packs logic into the logic cells, but will always leave the original nets in the design intact; what you see on your schematic is what you get. This approach will take up more logic resources and produce lower performance, so it should only be used when a design has

© 2008 QuickLogic Corporation

been fully hand-optimized. While the Level 1 Technology Mapper is the preferred logic optimizer in almost every case, Level 0 Optimization is provided for versatility and compatibility with older designs.

12.1.2 Level 1: Technology Map

The Level 1 Technology Mapper uses a sophisticated algorithm to perform logic optimization and efficient mapping of logic into logic cells. It will take advantage of the input registers and output OR gates whenever possible. Consequently, the Technology Mapper sometimes takes longer to run. For example, the Packer never takes more than a few seconds to run, but the Technology Mapper can take from a few seconds to several minutes depending on the complexity of the design.

© 2008 QuickLogic Corporation

12.2 Choosing Area or Speed Optimization

For all device families the Area/Speed option controls whether the Logic Optimizer uses the Timing Constraints specified in the Constraint Editor. The Speed setting uses these constraints to produce faster, but possibly larger results, while the Area option ignores timing constraints. See the **"Design Constraints and Analysis" on page 287** for more information on setting constraints.

NOTE: Internal nets may be deleted by the Level 1 or 2 Technology Mapper as the result of bubble-pushing, gate-collapsing, and other optimization methods. This will be noticeable in back-annotated (timing) simulation. Deleted nets will not be viewable in the simulator. The Report File (`design.rpt`) contains a list of nets deleted by Logic Optimization.

For more information, see also:

- **"Timing Constraints" on page 288** - Speed Optimization Using Timing Constraints
- **"Placement Constraints" on page 296** - Area Optimization Using Fixed Placement

12.3 Logic Optimization Modes

To choose an Optimization Mode, select **Tools>Options>Logic Optimizer**. The options are for Level 1 optimization only. Level 0 optimization is a simple, predictable algorithm which does not require different modes.

Preliminary Level 1 optimization completes in half the time of Quality Mode, with slightly optimal results.

Quality is the default mode of optimization; it is used for high quality results.

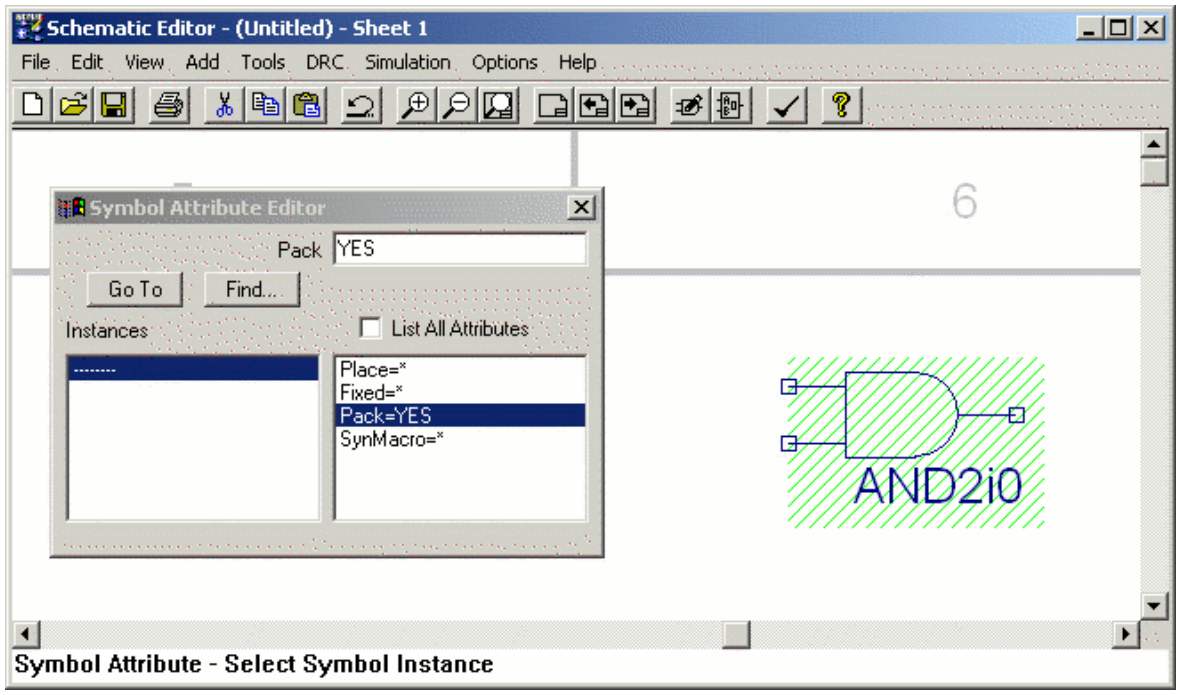
Overnight (or Exhaustive) mode produces slightly better results than Quality mode on some designs, but with a significantly longer run time (possibly several hours).

© 2008 QuickLogic Corporation

12.4 Using the Pack Attribute on Instances in a Design

Although many designs can benefit from the use of the Level 1 Technology Mapper, many macro library components are tuned for the Level 0 Optimizer (the Packer). Therefore, a Pack attribute has been implemented.

To assign a Pack attribute on a component instance, in the **SCS Schematic Editor**, select **Edit>Attribute>Symbol Attribute**. By making the attribute assignment **Pack=YES** on an instance (a schematic component), the optimizer knows to always perform Level 0 optimization on that component (including any sub-components).



The above figure illustrates the assignment of a `Pack=YES` attribute on a component instance using schematic design entry.

To tell the Level 1 Technology Mapper to ignore the `PACK` attribute placed in your design, select **Tools>Option>Logic Optimizer**. Check the **Ignore PACK attribute** option box when you want to evaluate the density and performance of a design with and without the use of the `PACK` attribute.

© 2008 QuickLogic Corporation

12.5 Automatic Buffer Insertion

Automatic buffer insertion applies to all device families.

When a design has high fanout nets, as reported by the Verifier upon import of the QDIF or the EDIF file, selecting the Auto Buffer Insertion option will tell the Logic Optimizer to utilize un-used logic cells to buffer the high fan-out nets. The Logic Optimizer will use up to, but never exceed, 100% of the logic resources available. The buffers will be inserted after the Placer has been run, so the buffering will have the logic placement in mind, resulting in better performance. The Automatic buffer insertion feature is especially important for synthesized HDL designs, as inserting buffers in the HDL source code is often not possible, nor desired. Utilizing the auto buffer insertion feature can get your design up to speed with a single click of your mouse button.

© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation

Chapter 13
Placer

The main task of the Placer is to take a design in the form of QuickLogic Logic Fragments (from the *Logic Optimizer*) and to place these Logic Fragments in optimal locations on the chip. The Placer determines optimal locations by looking at timing constraints (optional) added by the designer in the *Static Timing Analyzer* (see section **"Path Constraint-Driven Placement" on page 149**) or in the *Constraint Editor*, and by looking at the nets connecting logic cells together. Three placement modes are available in SpDE: *Preliminary*, *Quality*, and *Overnight*.

This chapter addresses the following subjects:

- **"Window-Based Placer" on page 143**
- **"Placer Options" on page 147**
- **"Path Constraint-Driven Placement" on page 149**
- **"Fixed Placement" on page 151**
- **"Timing-Driven Placement" on page 154**
- **"Low Power Placement" on page 154**

13.1 Window-Based Placer

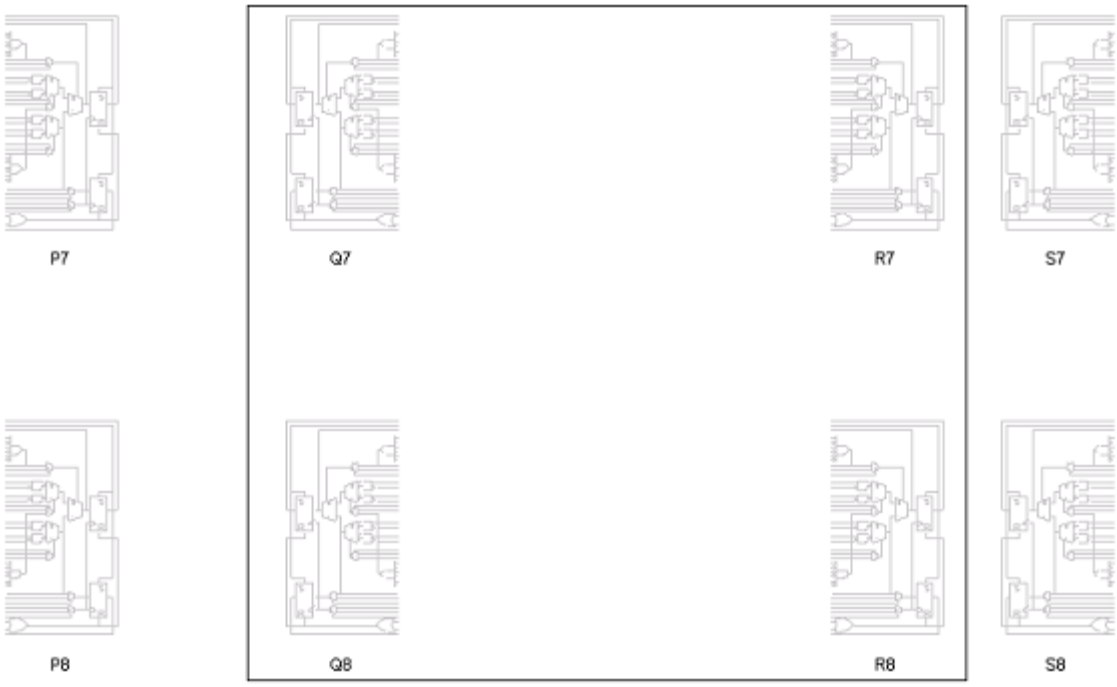
The QuickLogic Window-Based Placer offers the flexibility to place various modules of a design close together or far apart, so that the intended design specifications are met. The function provided by the Window-Based Placer is commonly referred as *floorplanning* in ASIC design. The Window-Based Placer, the same as floorplanning, offers the designer the flexibility to place two or more logic blocks or modules close to each other in order to conserve space and achieve timing requirements. Giving users control over placement of various logic blocks or modules assures that the designer's knowledge of the design structure is exploited to meet/exceed the chip space and performance threshold. This feature might not be useful for all designs, but is advantageous for those designs where it is crucial to improve critical path delays and/or to use the silicon area wisely.

The Window-Based Placer is activated through a constraint specified in the `.qcf` file. This `.qcf` file is the QuickLogic Constraint file, where pin constraints, flip-flop placement, timing constraints, etc. are usually given. A module can be placed in a pre-selected region of the chip, called a window, which is specified using the upper-left corner logic cell name and the lower-

© 2008 QuickLogic Corporation

right logic cell name. For example, as shown in **Figure 13-1**, if a 4-bit counter needs to be placed using logic cells Q7, R7, Q8, and R8, then the upper-left corner logic cell name would be specified as Q7 and the lower-right corner logic cell name as R8.

Figure 13-1: Placing Constraints



The command format to place a module in the design is:
window_in_placer <modulename> < LogicCell1> <LogicCell2>.

Using the previous 4-bit example, if this module was called count4, the command syntax to place in window Q7:R8 is:

window_in_placer count4 Q7 R8

Window-Based Placer characteristics:

- The ability to place all the logic in the design in one condensed area is done by using the ALLGATES keyword—this keyword is not case sensitive.
  Example:
  window_in_placer ALLGATES a1 am53
  This will place all the gates in the design inside the [a1 am53] window.
- When a window is too small to hold the module, Window-Based placer will provide the estimated number of logic cells needed for the window to accommodate the module and will let the user change the window on the fly. SpDE will give the following error message: "WindowBasedPlacer: Window(A1, F6) is too small (121 logic cells needed). Please increase window size and re-run the placer".
- All windows specified must be rectangular with the boundaries specified by the upper-left corner logic cell address and lower-right corner logic cell address.The format for specifying such a window is:
  window_in_placer <modulename> < LogicCell1> <LogicCell2>

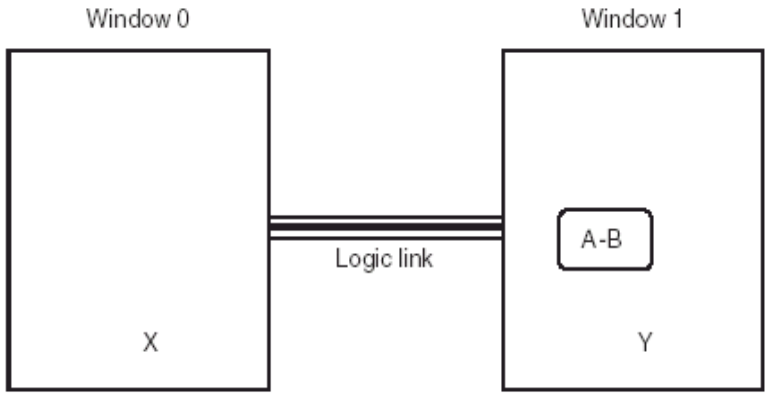
© 2008 QuickLogic Corporation

where <LogicCell1> refers to the upper-left corner logic cell address and <LogicCell2> refers to the lower-right corner logic cell address. In the example above the Window Placer has the following characteristics:

- window_in_placer is case sensitive.
- The module name must match the name given in the HDL or schematic file and is case sensitive.
- The logic cell address is not case sensitive.

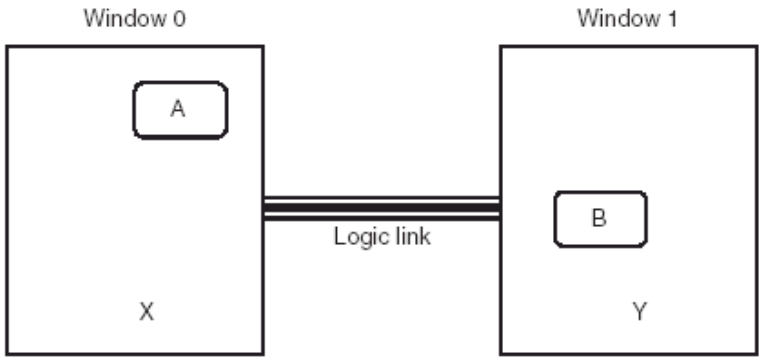
- Each module must be associated with only one window. The following is *NOT* allowed:
  - **window_in_placer** I2 a1 k12
  - **window_in_placer** I2 ak1 au12
- Splitting modules into two windows is not supported. If the above constraints were placed, only the first window (a1 k12) would be used, the second would be ignored. The following is allowed:
  - **window_in_placer** I2 a1 k12
  - **window_in_placer** I3 a1 k12
    Placing two modules in one window or windows overlapping is allowed
- Placement of flip-flops in a window that is already assigned to a module is allowed. For example, the following is allowed:
  - **window_in_placer** I6 aj50 am53
  - **ffplacement** I6.n_11 al53
    In this case, although window [aj50 am53] is already assigned to module I6, the flip-flop with I6.n_11 as its output is fix-placed in logic cell al53, which is inside the [aj50 am53] window.
- If gates A and B are defined inside different modules, X and Y respectively, and are optimized or combined during Technology Mapping, the combined A-B logic will be placed either in window X or Y (**Figure 13-2 on page 145**). The reverse can also occur where a large combinatorial block in one module/window is split between two windows (**Figure 13-3 on page 146**).

Figure 13-2: A-B Logic Placed in Either Window



The diagram in **Figure 13-2 on page 145** could be logically optimized during Technology Mapping to the diagram shown in **Figure 13-3 on page 146**.

Figure 13-3: Combinatorial Block in One Module/Window is Split Between Two Windows



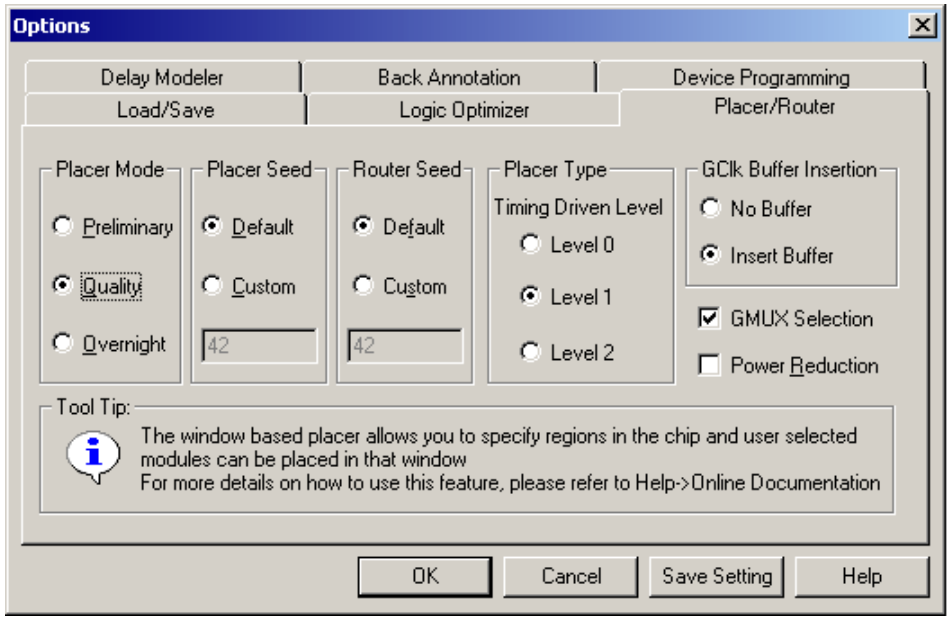
For more information, see also:

- Windows Based Placer - Application Notes [PDF]
- **"Window-Based Placement Editor" on page 280** - Using Windows Based Placer Editor

© 2008 QuickLogic Corporation

13.2 Placer Options

Access the Placer options from **SpDE**, by selecting **Tools>Options**.



13.2.1 Placer Seed

A placement seed is used to initialize the placement process and set a starting point for the decisions made during automatic placement. SpDE allows the user to specify a custom seed, or to use the default seed. The specific value of a seed has no meaning whatsoever. What matters is that changing the seed sets a completely different starting point for the placer, which can produce a slightly different placement. QuickLogic has chosen a specific default seed value of `42` for important historical (although completely meaningless) reasons.

13.2.2 Placer Mode

The Placer can be run in one of three placement modes, each with different characteristics:

- **Preliminary**—placement is faster than quality placement. However, the results are not as predictable, and usually not as optimal, as with the Quality placement. QuickLogic recommends that a design should be placed in at least Quality mode before a chip is programmed.
- **Quality**—the default placement mode; it produces high quality placements. The difference between Quality and Overnight placement is very small (a statistical average of 4% improvement in Overnight mode). However, Quality placement is much faster than the Overnight placement.
- **Overnight**—the results can be about 4% better than using Quality mode, but it takes about ten times longer to run.

© 2008 QuickLogic Corporation

Table 13-1 on page 148 shows the relative time impact of choosing different Placer modes.

Table 13-1: Example of Placement Times

Placement Mode	Run Time
Preliminary	x (example: 1 minute)
Quality	6x (example: 6 minutes)
Overnight	60x (example: 1 hour)

13.2.3 Placer Type

The Placer can be run in one of three placement levels:

- **Level 0** placement ignores the timing constraints entered in the Constraints Editor (**Tools/Constraint Manager/Timing**).
- **Level 1** placement is the default placement type. It does the placement in such a way so that the timing constraints entered in the Constraints Editor (**Tools/Constraint/Timing**) are met. Compared with the results of the Level 0 placement, the Level 1 placement produces much higher quality placements in terms of clock timing performance.
- **Level 2** placement also works with the timing constraints. Its runtime is much longer than the Level 1 placement. It may occasionally produce slightly better results.

13.2.4 Placer with Low Power

The Placer can run with a power saving option, if the **Power Reduction** box is selected. It will try to minimize the power consumption of the device during the placement process.

13.2.5 Gclk Buffer Insertion

The Global Clock Buffer Insertion option is used to decide whether the global clock buffers should be inserted for nets whose loads are spread across multiple quandrants and high fanout nets.The default option for the buffer insertion is insert buffers.

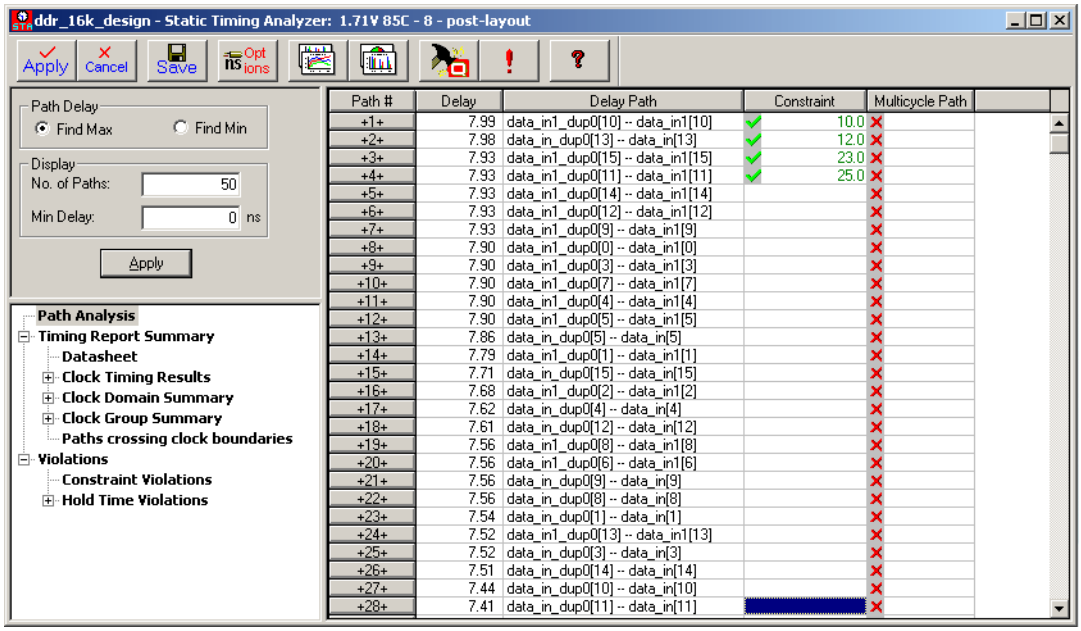
© 2008 QuickLogic Corporation

13.3 Path Constraint-Driven Placement

Timing-driven placement allows the user to obtain peak performance without resorting to fixed placements. The Placer works closely with the Path Analyzer to provide *path constraint-driven placement*—user-specified constraints are fed to the Placer; paths not meeting the specified constraints are automatically boosted in priority.

Constraints can be entered for these paths directly in the Path Analyzer. Once the Path Analyzer has run, paths not meeting the desired goal can be easily identified. For more information on setting design constraints, see the chapter **"Design Constraints and Analysis" on page 287**.

NOTE: The Placer's speed optimization based on the constraints specified in the Path Analyzer is independent of the placement type (i.e., levels of placement).


ddr_16k_design - Static Timing Analyzer: 1.71V 85C - 8 - post-layout
Apply
Cancel
Save
Options
Path Delay
Find Max
Find Min
Display
No. of Paths: 50
Min Delay: 0 ns
Apply
Path Analysis
Timing Report Summary
Datasheet
Clock Timing Results
Clock Domain Summary
Clock Group Summary
Paths crossing clock boundaries
Violations
Constraint Violations
Hold Time Violations
Path # | Delay | Delay Path | Constraint | Multicycle Path
+1+ 7.99 data_in1_dup0[10] -- data_in1[10] 10.0
+2+ 7.98 data_in_dup0[13] -- data_in[13] 12.0
+3+ 7.93 data_in1_dup0[15] -- data_in1[15] 23.0
+4+ 7.93 data_in1_dup0[11] -- data_in1[11] 25.0
+5+ 7.93 data_in1_dup0[14] -- data_in1[14]
+6+ 7.93 data_in1_dup0[12] -- data_in1[12]
+7+ 7.93 data_in1_dup0[9] -- data_in1[9]
+8+ 7.90 data_in1_dup0[0] -- data_in1[0]
+9+ 7.90 data_in1_dup0[3] -- data_in1[3]
+10+ 7.90 data_in1_dup0[7] -- data_in1[7]
+11+ 7.90 data_in1_dup0[4] -- data_in1[4]
+12+ 7.90 data_in1_dup0[5] -- data_in1[5]
+13+ 7.86 data_in_dup0[5] -- data_in[5]
+14+ 7.79 data_in1_dup0[1] -- data_in1[1]
+15+ 7.71 data_in_dup0[15] -- data_in[15]
+16+ 7.68 data_in1_dup0[2] -- data_in1[2]
+17+ 7.62 data_in_dup0[4] -- data_in[4]
+18+ 7.61 data_in_dup0[12] -- data_in[12]
+19+ 7.56 data_in1_dup0[8] -- data_in1[8]
+20+ 7.56 data_in1_dup0[6] -- data_in1[6]
+21+ 7.56 data_in_dup0[9] -- data_in[9]
+22+ 7.56 data_in_dup0[8] -- data_in[8]
+23+ 7.54 data_in_dup0[1] -- data_in[1]
+24+ 7.52 data_in1_dup0[13] -- data_in1[13]
+25+ 7.52 data_in_dup0[3] -- data_in[3]
+26+ 7.51 data_in_dup0[14] -- data_in[14]
+27+ 7.44 data_in_dup0[10] -- data_in[10]
+28+ 7.41 data_in_dup0[11] -- data_in[11]

NOTE: It is important to set the constraints realistically—set each constraint at or just slightly below the required value. One of the keys to timing-driven placement is the concept of *good enough*. Once a critical path has met its constraint, the Placer will boost the priority elsewhere in order to optimize all critical paths.

For each path with a constraint, the Placer estimates the delay throughout the placement process. If a constraint is met, the placer will continue to optimize the nets in the path normally. If a constraint is not met, the placer will boost the priority of the nets in the paths; the boost in priority is proportional to the difference between the constraint and the estimated value. In other words, paths near their constraints will be boosted in priority less than paths far from their constraints.

NOTE: Add constraints only where required. The dynamic delay estimation mentioned above adds work to the placement process. Each constraint specified will slow the placement process.

Path constraints are stored in the QuickLogic Constraint File (`.QCF`). Once a constraint has been entered, the `.QCF` file is automatically created and updated.

© 2008 QuickLogic Corporation

NOTE: Constraints should be edited in the Static Timing Analyzer, not in the constraint file. Deleting the constraint file will remove all constraints.

In the Path Analyzer option dialog box, if the **Always show constrained paths** checkbox is selected, the constrained paths will always be listed in the Path Analyzer, even if you choose a set of paths that would not normally consist of the constrained paths. All subsequent Placer runs will operate in timing-driven mode. This can be verified during placement from the SpDE Status window—under normal placement the heading is Placer, whereas under timing-driven placement the heading is Timing-Driven Placer.

NOTE: To delete an existing constraint(s) from the Path Analyzer, select the constraint or constraints with the mouse, then click **Delete**.

For more information, see also:

- Constraint File Format (.QCF File) [PDF]
- **"Static Timing Analyzer" on page 319** - Overview
- **"Static Timing Analyzer Options Pane" on page 321** - Using the Path Analyzer Tool
- **"Path Analysis" on page 322** - Overview
- **"Path Analyzer Options" on page 323** - Using the Path Analyzer Tool
- **Chapter 22, "Design Constraints and Analysis" on page 287** - Applying Various Design Constraints

© 2008 QuickLogic Corporation

13.4 Fixed Placement

Although the Placer will automatically determine placement for logic cells and I/O pads, it also supports fixed assignment of both I/O and flip-flops when required. As the name implies, fixed assignments made by the designer are not modified by the Placer.

The easiest way to fix pins and flip-flops is by using the Placement Constraint tool. In SpDE, select **Tools>Constraint Manager**. For more details see the chapter **"Design Constraints and Analysis" on page 287.**

For more information, see also:

- **"Using Constraint Manager" on page 268**
- **"Placement Editor" on page 296** - Using Placement Editor Tool

13.4.1 Fixing the Placement of I/O Pads

Design constraints sometimes require some or all I/O cell locations to be fixed. For example, an existing printed circuit board (PCB) might dictate a precise pinout. Alternatively, a high-speed PCB might require fixing a small number of critical pins in order to limit skew. The SpDE Placer can handle these cases.

I/O cell locations are fixed using an attribute assignment within the design hierarchy. There are two attributes which may be assigned for each I/O pad: **PLACE** and **FIXED**. The PLACE attribute is normally left blank (for automatic placement) but it may be set to the pin number where the I/O pad should be placed. If fixed placement is needed, the FIXED attribute should be set to **YES**. (Setting the FIXED attribute to NO tells the Placer to ignore the PLACE attribute.) If the PLACE attribute is given a value, then FIXED defaults to **YES** if left blank.

13.4.1.1 Manually Assigning Pin Placement in QuickWorks Schematic-Based Designs

In the SCS schematic editor, attributes can be added by selecting **Edit>Attribute>Symbol Attribute**. Click on the component to add the **PLACE** or **FIXED** attribute.

Attributes can also be added in the Hierarchy Navigator by selecting **Edit>Attribute> Symbol Attribute**. Attributes applied in the Navigator take precedence over those applied in the schematic editor. Also, when you apply attributes in the Navigator, select **File>Save** to save these attributes in the Navigator tree (`.TRE`) file so that they are not lost when you close the Navigator.

NOTE: Instance and net names should be supplied throughout the design. User-specified instance and net names will ensure that I/O cells and flip-flop locations are properly back-annotated if the design is modified.

Keep in mind that two I/O macros cannot be assigned to the same pin. Furthermore, high-drive macros (HDPAD, HDiPAD, HDdPAD, HDPAD_25 um) must be assigned to input I cells and clock macros (CKPAD, CKdPAD, CKtPAD, CKPAD_25 um) must be assigned to clock (ACLK, GCLK, or CLK) pins. All other I/O macros must be assigned to bi-directional (I/O)

© 2008 QuickLogic Corporation

cells. For pinout information, refer to the appropriate device datasheets available from the QuickLogic Web site [www.quicklogic.com]. SpDE verifies all of these rules with the Design Verifier (runs as soon as a QDIF file is imported to SpDE).

13.4.1.2 Manually Assigning Pin Placement in QuickWorks Verilog- or VHDL-Only Designs

In QuickWorks an entire design can be entered using Verilog or VHDL (no schematics). In many cases it is necessary to be able to fix pin locations and make special assignments. When describing a complete design in Verilog or VHDL this is accomplished in a separate file with the `.sc` extension, rather than in the Verilog or VHDL source file itself. This can be done with the following syntax:

```
#---Fixed I/O cells---
portprop ld ql_placement = “io15”;
portprop clk ql_placement = “io17”, ql_padtype=“CLOCK”;
portprop d[1] ql_placement = “io22”;
portprop d[0] ql_placement = “io12”;
```

Notice that pin numbers must be prefixed with `“io”`. (This is not required for a BGA package.) Also notice that for the signal named `clk` in the third line above, a `ql_padtype` property was added to indicate that a `CLOCK` pad should be used to access internal clock networks. The legal `ql_padtypes` are listed in **Table 13-2 on page 152**.

Table 13-2: ql_padtypes

INPUT	Use this for HDPADs (pins marked with I on the pinout table)
CLOCK	Use this for CKPADs (pins marked with CLK, GCLK/ I, or ACLK/ I)
BIDIR	Normal I/O pad (pins marked with I/O)
CLOCKB	Use this for CKPADP5 (clock pin for QL5064)

13.4.1.3 Fixing the Placement of Flip-Flops

Design constraints rarely require logic cell locations to be fixed. However, to allow the designer a greater degree of flexibility, the Placer allows some or all of the flip-flop macros to be fixed. One scenario that would dictate fixed flip-flops would be where all the bits of an 8-bit register need to appear on the output pins with absolute minimum skew. The Placer, not realizing this design constraint, might sacrifice the skew on the outputs in order to produce an overall faster circuit. By manually fixing the flip-flops on logic cell locations adjacent to the output pins, the designer can meet the design constraint.

13.4.1.3.1 Fixing Flip-Flops Manually in QuickWorks Schematic-Based Designs

Logic cell locations can be fixed in QuickWorks in the same way that `I/O` pins are fixed by using a symbol attribute assigned to a component (also known as an *instance*) in the design. Refer to the previous section on fixing `I/O` pads for details.

NOTE: Instance and net names should be supplied throughout the design. User-specified instance and net names will ensure that `I/O` cells and flip-flop locations are properly back-annotated if the design is modified.

© 2008 QuickLogic Corporation

Although the *Place* attribute must be assigned to a flip-flop macro, the fixed placement affects the entire logic cell containing the flip-flop. For example, if a multiplexer and a pair of AND gates are packed with the flip-flop (thereby mapping into a single logic cell), these macros will be fixed as a unit. In this regard, the Place attribute is assigned not just to the flip-flop, but to all macros packed into the flip-flop's logic cell.

Keep in mind that two flip-flop macros cannot be assigned to the same location. The naming convention for logic cells assigns a character to each column and a decimal number to each row. The logic cells in a `QL1P100`, for example, are named `A1` through `AF1` in the first row, and `A20` through `AF20` in the final row. SpDE verifies all of these rules with the *Design Verifier*.

13.4.1.3.2 Fixing Flip-Flops and I/O Pins Automatically

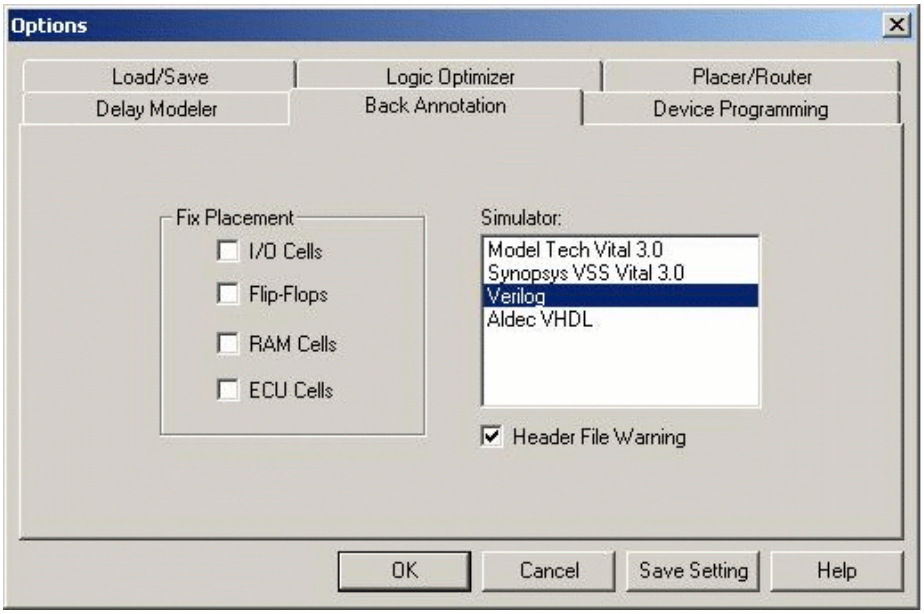
Take the following two steps to fix pins or flip-flops in QuickWorks designs:

1. Select **Tools>Options**. Select the appropriate options in the **Back Annotation** tab, then click OK.
2. To run the Back-Annotation tool, in **SpDE**, select **Tools>Run Selected Tools**. This step creates the `.QCF` file needed to fix pins and flip-flop locations for all future place and route iterations.

13.4.1.4 Locking Down a Previous Placement

It is sometimes necessary to lock down an `I/O` placement or a logic cell placement. This means that for all subsequent place and route runs, you do not want the `I/O` pins to change their locations and/or you do not want the logic cells to change their locations.

The Placement editor is the best tool to use to fix a subset of the flip-flops or pins in your design. For details see **"Design Constraints and Analysis" on page 287**.To fix the placement of all the pins and/or all the flip-flops, you must use the Back Annotation tool. To tell this tool to fix `I/O` or logic cell placements, set the tool options by selecting **Tools>Options** in SpDE.



In the **Back Annotation** tab, check the **I/O Cells** box to force the `I/O` pads to be locked into their current position for subsequent design iterations. Similarly, select Flip-Flops, RAM Cells, or ECU Cells to force the locations of the flip-flops, RAM Cells, or ECU Cells in the design to remain fixed through subsequent design iterations.

13.5 Timing-Driven Placement

When the placement type is set to Level 1 or Level 2, the Placer performs timing optimization based on the timing constraints entered in the Constraints Editor (**Tool>Constraint Manager>Timing**). The Placer estimates the delay throughout the placement process. The *slack* (i.e., the timing difference between the required delay and the actual current delay) of each net based on the specified constraint is calculated, and the placer boosts the priority of the nets according to the value of their slack. The boost in priority is proportional to the value of the slack. If the timing constraints are not specified, the Placer still estimates the delay, and calculates the slack for each net with the constraint set to be the delay value of the longest path in the corresponding clock domain.

The timing performance yielded by the timing-driven placement is much better (a statistical average of 20% improvement) than the non-timing-driven placement (Level 0). The difference between path-constraint-driven placement and the timing-driven placement is that path-constraint-driven placement works on the user-specified set of paths and tries to make them meet the constraints, while timing-driven placement tries to meet the timing constraint by working on the nets with large slack values. For timing-driven placement, the user need only to set the constraint for each clock; while for path-constraint-driven placement, the user has to specify constraint for all paths that do not meet the constraint. For timing critical designs, the best timing performance can be achieved by using these two techniques at the same time.

For more information, see also:

- **“Using Constraint Manager” on page 268**
- **“Timing Constraints” on page 288** - Using Timing Constraints Editor Tool

13.6 Low Power Placement

When the **Power Reduction** option is selected, the Placer gives high priority to reduce the power consumption of the design by reducing the number of column clock buffers used by the design. However, the timing performance will be affected in this mode.

© 2008 QuickLogic Corporation

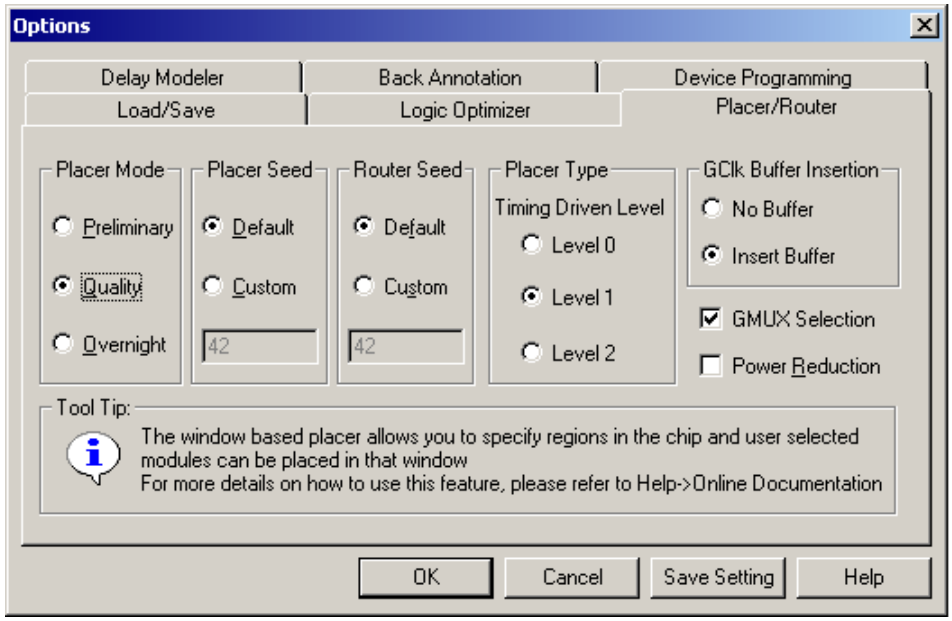
Chapter 14
Router

The *Router* employs highly optimized algorithms to connect I/O and logic cells using the pASIC interconnect resources. This finely tuned arrangement produces excellent performance with high utilization. This chapter addresses the following subjects:

- **“Router Tool Options” on page 155**
- **“Using the Priority Router” on page 156**
- **“Constraint-Driven Routing” on page 157**
- **“Interconnect Resources” on page 158**
- **“Routing High-Drive Nets” on page 162**
- **“Manual Routing Editor” on page 163**

14.1 Router Tool Options

To change the setting for the Router Seed in SpDE, select **Tools>Options**.



14.1.1 GMUX Selection

The GMUX Selection option allows the user to decide whether the nets in the design should be routed through the Global MUX with user selection. The default option is FALSE.

© 2008 QuickLogic Corporation

14.2 Using the Priority Router

The Priority Router enables the user assign priorities on the selected critical nets that are specified to be routed before other nets. The critical nets can be routed so that the wire delay of the nets in the critical path is minimized. The routing priority of the nets can be specified as a constraint in the `.qcf` file. There are two interfaces for specifying the priorities constraints:

- `.qcf` file interface
- Static Timing Analyzer interface

14.2.1 QCF File Interface

The `.qcf` file supports the routing priority of nets specified as constraints. If the priority of nets is known before hand, the same priority can be specified through `.qcf` file using following syntax:

```
# net routing priority constraints
routepriority <net_name> <priority_level>
```

routepriority—keyword indicating that the constraint specifies net route priority
net_name—net that should be prioritized
priority_level—integer value in the range of 0-2 where:
0 is Normal Default Priority
1 is High Priority
2 is Very High Priority
(e.g., routepriority PCI.MASTER.DATA[0] 2)

For more information, see also:

- Constraint File Format (.QCF File) [PDF]

14.2.2 Static Timing Analyzer Interface

The user can identify the nets that are part of the critical paths and change the routing priority of these through the Static Timing Analyzer viewer pane.

To view and change the priority of nets, click on the new **Show Routing Priority** icon in the Static Timing Analyzer results window. There are three more columns visible in this mode:

- **Net Name**—name of the net whose routing priority is specified.
- **Number of Paths**—number of paths (out of listed in Path Analyzer window) that depend on this net.
- **Routing Priority**—current routing priority of the net. The routing priority can be changed from here.

To run the Path Analyzer:

1. Open the Path Analyzer by using the STA toolbar button or by selecting **Tools> Static Timing Analyzer>Path Analysis** from the SpDE menu bar. The Path Analyzer Options dialog box appears.

© 2008 QuickLogic Corporation

2. Click on **Run**. The paths are listed in the Static Timing Analyzer viewer pane according to the start and stop set selection.

For more information, see also:

- **"Static Timing Analyzer" on page 319** - Overview
- **"Static Timing Analyzer Options Pane" on page 321** - Using the Path Analyzer Tool
- **"Path Analysis" on page 322** - Overview
- **"Path Analyzer Options" on page 323** - Using the Path Analyzer Tool

14.2.3 Running Router Tool

If the net route priority constraints are added either through the manual edit of .qcf file or through the Static Timing Analyzer, at the next run the router automatically uses them to guide the routing. There is no special setting required for forcing the router to use these constraints.

NOTE: Any manual changes on the `.qcf` file will only be applied during the next load of the design.

14.2.4 Router Tool Limitations

- Ripping priority nets during Routing phase. Most of the time, the prioritized nets are not ripped once they are routed. In some cases, where it is necessary, the router tries to rip a low priority net before ripping a high priority net. In few cases, high priority nets might get ripped out of order without really differentiating them from non-prioritized nets.
- If the net priority value is changed from Static Timing Analyzer, the same will be applied immediately. Clicking on **Cancel** button of the Static Timing Analyzer toolbar does not undo the changes already done. This undo limitation is only for net route priority changes.

14.3 Constraint-Driven Routing

The Router will optimize the routing of critical nets when timing constraints are specified with the Constraint Editor. For details see **"Design Constraints and Analysis" on page 287**.

© 2008 QuickLogic Corporation

14.4 Interconnect Resources

The QuickRAM and QuickPCI family of devices from QuickLogic contain the same types of routing resources as the pASIC3 family, with some additions. The pASIC3, QuickRAM, and QuickPCI families add dual wires and feature two different types of clock networks.

14.4.1 Array Clock Networks

The Array Clock networks are simply called *clock networks* in the pASIC3 devices. They are driven by the dedicated input pins marked ACLK/ I in the pASIC3, QuickRAM, and QuickPCI device pinout diagrams.

The Array Clock networks are dedicated resources that are capable of connecting to the clock, set, or reset of any array flip-flop in the pASIC3, QuickRAM, and QuickPCI devices, but they may NOT connect to combinatorial logic or any of the I/O pads or input registers. Each Array Clock network must be driven by one of the clock pads in the macro library: CKPAD, CKtPAD, CKPADff, CKtPADff.

NOTE: The CKPADff and CKtPADff macros can only be used in pASIC3, QuickRAM, and QuickPCI designs.

When creating a Verilog or VHDL design, the syn_isclock attribute should be set to CLOCK. Specific pins must be used depending on the package type. For the pinout tables for pASIC3, QuickRAM, and QuickPCI devices, refer to the appropriate device datasheets available from the QuickLogic Web site [www.quicklogic.com].

14.4.2 Global Clock Networks for Eclipse and EclipsePlus Devices

The QuickLogic Eclipse devices contain nine global clock networks (one dedicated and eight programmable clock networks).

The Eclipse dedicated clock connects to the clock input of the logic cell, I/O, and RAM registers through a hard-wired connection and is multiplexed with the programmable clock input. The dedicated clock provides a fast global network with low clock skew and has access to one of the four Phase Lock Loops (PLLs); it is connected to only one clock pin in the Eclipse and QuickLogic devices. The dedicated clock is driven by the dedicated input pin marked as CLK DEDCLK/ PLLIN in the Eclipse and EclipsePlus device pinout diagrams.

The Eclipse eight global clock networks can drive clock inputs of logic cells, I/Os, Embedded Computational Units (ECUs), and RAM registers. The global clocks can also drive the F1 and A2 inputs of the logic cells. Three global clocks will each have access to a separate PLL. The dedicated input pins marked as CLK/ PLLIN in the Eclipse and EclipsePlus device pinout diagrams drive these three global clocks. The clock pad in the PAD macro library used to drive global clocks for Eclipse and EclipsePlus devices is CKPAD_25 um.

There are five Quad-Net local clock networks in each quadrant for a total of 20 in each Eclipse device. Each Quad-Net local clock network is local to a quadrant. Quad-Net is multiplexed with the clock buffer before driving the column clock buffers. Each of the other five global clocks will have access to one of the five Quad-Net local clock networks with programmable connections to the register inputs. The dedicated input pins marked as CLK in the Eclipse device pinout diagrams drive these five global clocks. The GCLKBUFF_25 um pad macro is used to place any internally generated signal on the global clock network. If the number of clock

© 2008 QuickLogic Corporation

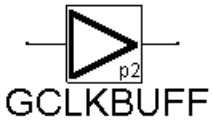
loads (fanouts) driven by the GCLKBUFF_25 um pad macro fits in just one quadrant of the Eclipse devices, the Quad-Net local clock network (local to the quadrant) is used. Otherwise, the global clock network is used—this prevents global clock resources being used when a Quad-Net will suffice.

When creating a Verilog of VHDL design, set the ql_padtype attribute to clock in the synthesis command (.sc) file. Specific pins must be used depending on the package type.

14.4.3 Global Clock Networks

The Global Clock networks complement the Array Clock networks in each pASIC3, QuickRAM, and QuickPCI devices. The Global Clocks may be driven by a clock pad placed on a `GCLK/I` pad location (see the pinout diagram for the device you are using in the appropriate datasheet available from the Quicklogic Web site [www.quicklogic.com]), or a global clock buffer inside the device (see **Figure 14-1**). The global clock buffer can be driven by any driver inside the device except an array or global clock network. This means that a global buffer can be driven by any I/O pad, any high-drive driver (such as an `HDPAD` or a high-drive output of a `CKtPAD`), or any internal logic. The global buffer cannot be driven by another global buffer, a `CKPAD`, or the middle driver of a `CKtPAD`; the global clock network cannot be driven by another clock network. For the best performance, use the dedicated input pin labeled: `GCLK/I` on the pinout diagram.

Figure 14-1: Global Clock Buffer Symbol

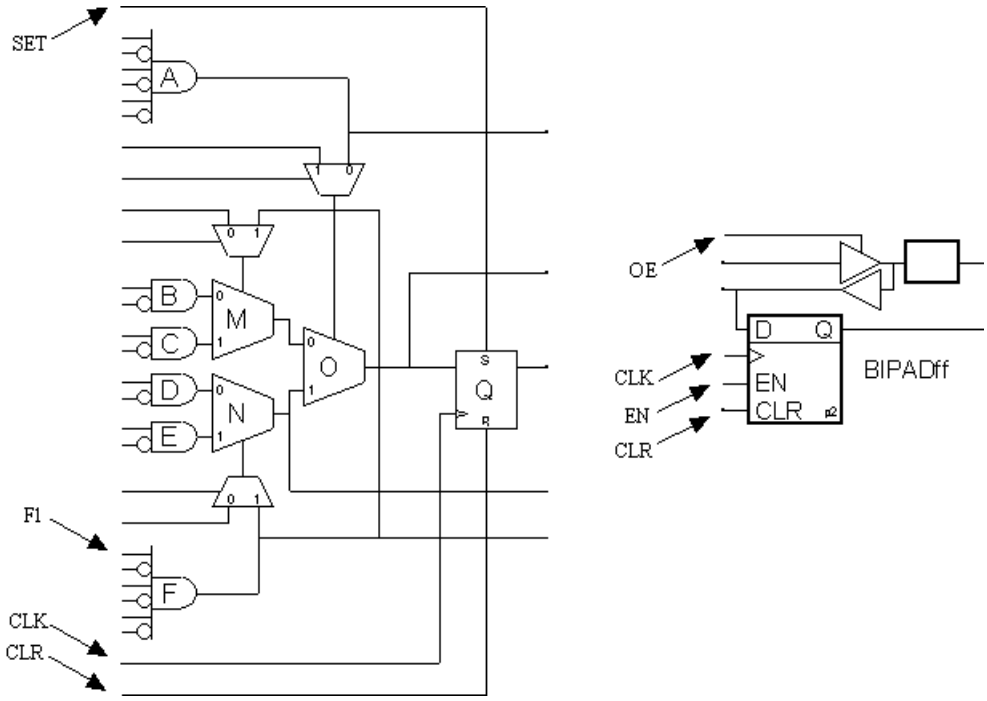


The Global Clock network can drive all the resources that the Array Clock is capable of driving (the clock, set, or reset of any array flip-flop). The Global Clock network can also drive the output-enable signals on I/O pads, as well as the input register control signals (clock, reset, and enable) and the `F1` input to the logic cells (refer to **Figure 14-2 on page 160**).

To drive the Global Clock network with one of the dedicated GCLK/I pins, you must use one of the clock pads in the macro library (`CKPAD, CKtPAD, CKPADff,` or `CKtPADff`). When creating a Verilog or VHDL design, set the `ql_padtype` attribute to `clock` in the synthesis command (`.sc`) file. Specific pins must be used depending on the package used.

If you drive the Global Clock network with internal logic or any other type of pad, you need to use the Global Clock buffer (GCLKBUFF) from the schematic library.

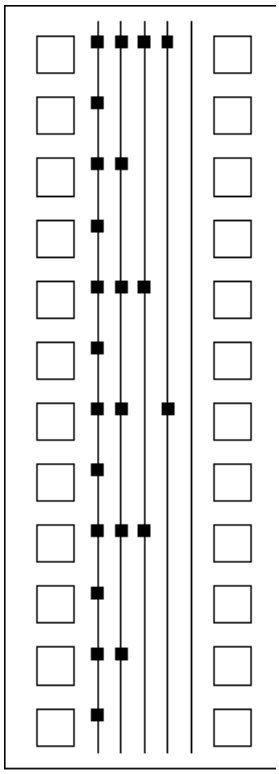
Figure 14-2: Global Clock Network connection points in pASIC3, QuickRAM, and QuickPCI devices



© 2008 QuickLogic Corporation

The Routing wires shown in **Figure 14-3** are (left to right): Segmented, Dual, Quad, and Express.

Figure 14-3: Routing Wires



14.4.3.1 Segmented Wires

Segmented wires are the shortest type of routing wires in the pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro, and PolarPro II devices. These wires traverse the distance of one logic cell. High-Drive pads cannot drive segmented wires, so the Router restricts nets on High-Drive pads to be routed on express, or quad wires.

14.4.3.2 Dual Wires

Dual wires exist only in the pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro, and PolarPro II family of devices. These wires traverse the distance of two logic cells. High-Drive pads cannot drive dual wires, so the Router restricts nets on High-Drive pads to be routed on express or quad wires.

14.4.3.3 Quad Wires

Quad wires span four times the distance the segmented wires span (four logic cells). They are present in the pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro, and PolarPro II devices.

© 2008 QuickLogic Corporation

14.4.3.4 Express Wires

Express wires are wires which span the entire length or height of the pASIC3, QuickRAM, QuickPCI, and Eclipse devices. They are used for high fanout nets or nets that need to travel across the device.

14.4.3.5 Distributed Network Wires

Distributed network wires are wires that span the programmable logic of the Eclipse devices, and are driven by *column clock* buffers. Each dedicated clock network pin buffer is hard-wired to a set of column clock buffers. Five global networks *global buffers* can be connected through special purpose routing called *HSCK lines* to either a dedicated pin buffer, or any vertical routing wire crossing it.

14.5 Routing High-Drive Nets

14.5.1 High-Drive Pads

High-Drive Pads (`HDPADs`, labeled as `I` in pinout diagrams, rather than `I/O, I/CLK, ACLK/I, or GCLK/I`) must drive either quad wires or express wires. In an HDL design, a high-drive input pad is specified with the `ql_padtype` attribute of `INPUT` in the `.sc` file.

14.5.2 Parallel Logic

The pASIC3, QuickRAM, QuickPCI, and Eclipse architectures allow quad or express wires to be driven by parallel logic. Parallel logic is a logic configuration where two identical gates (same inputs) have their output nets connected for higher drive capability. The type of gates that can be tied in 'parallel' is restricted. For more information on double-buffering, refer to **“Design Techniques” on page 121**.

NOTE: SpDE will warn you if you use more than the recommended limit of High Drive nets (nets driven by High-Drive Pads or Parallel Logic). The router may have difficulty completing successfully in these cases.

© 2008 QuickLogic Corporation

14.6 Manual Routing Editor

Routing generated for a given placement by the Router is generally of good quality but there could be a few nets that are loosely routed. The manual Routing Editor is an efficient tool to improve these loosely routed nets one after another.

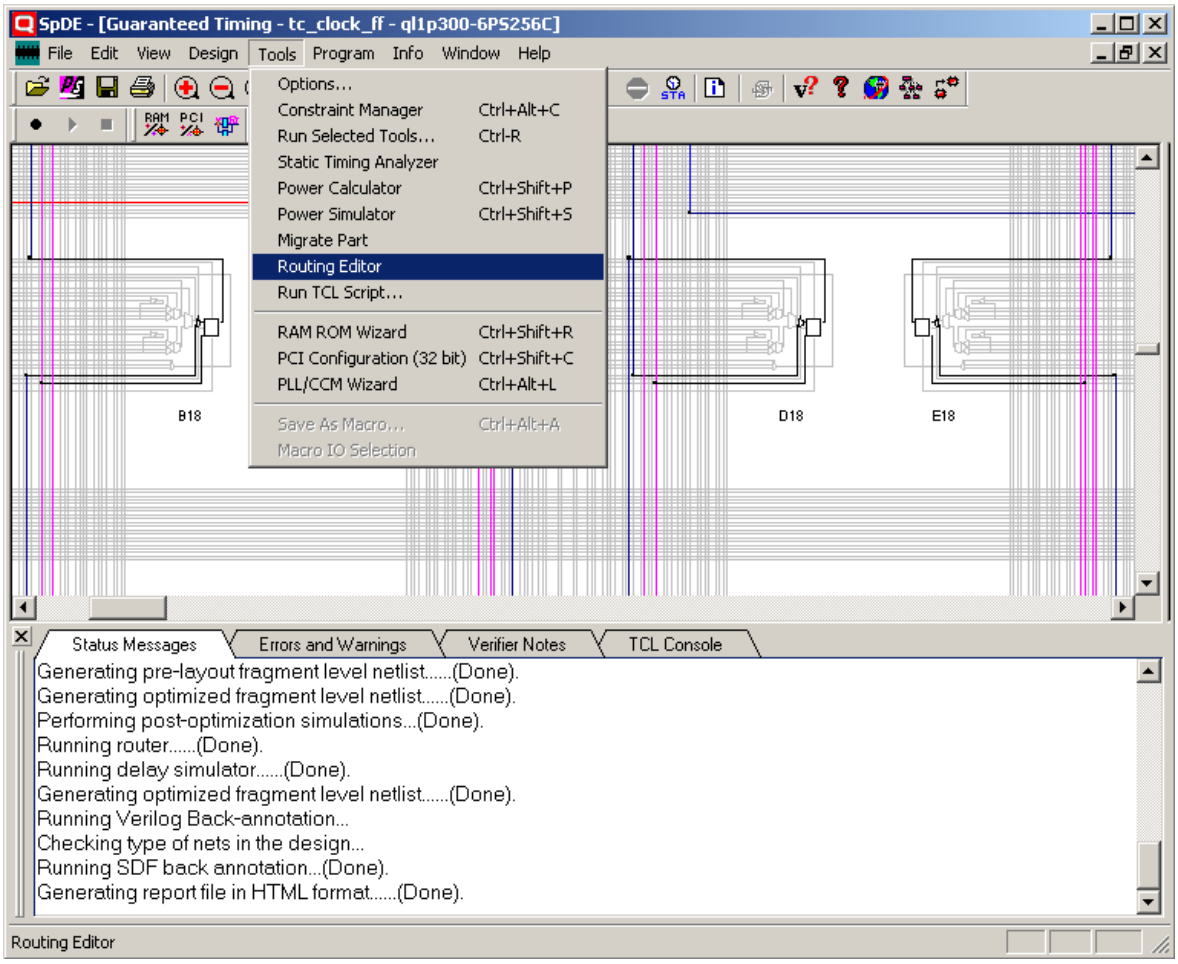
The following terminology is used throughout this section:

Loop—If a net crosses itself has a cross-link at the point where it crosses itself, it forms a *loop*. This is an invalid configuration for programming and the routing editor issues an error message.

Near-Loop—If a net crosses itself but does not connect with a cross-link at the cross-over point, it form a *near-loop*. This is an invalid configuration for programming and the routing editor issues an error message.

14.6.1 Opening Manual Routing Editor

The manual Routing Editor can be opened from SpDE by selecting **Tools>Routing Editor**.



NOTE: The manual Routing Editor is available only for routed designs.

14.6.2 Routing Editor Toolbar

When you open the Routing Editor, all nets are initially shown in black and a floating toolbar pops up. This toolbar contains five commands, as described in **Table 14-1**.

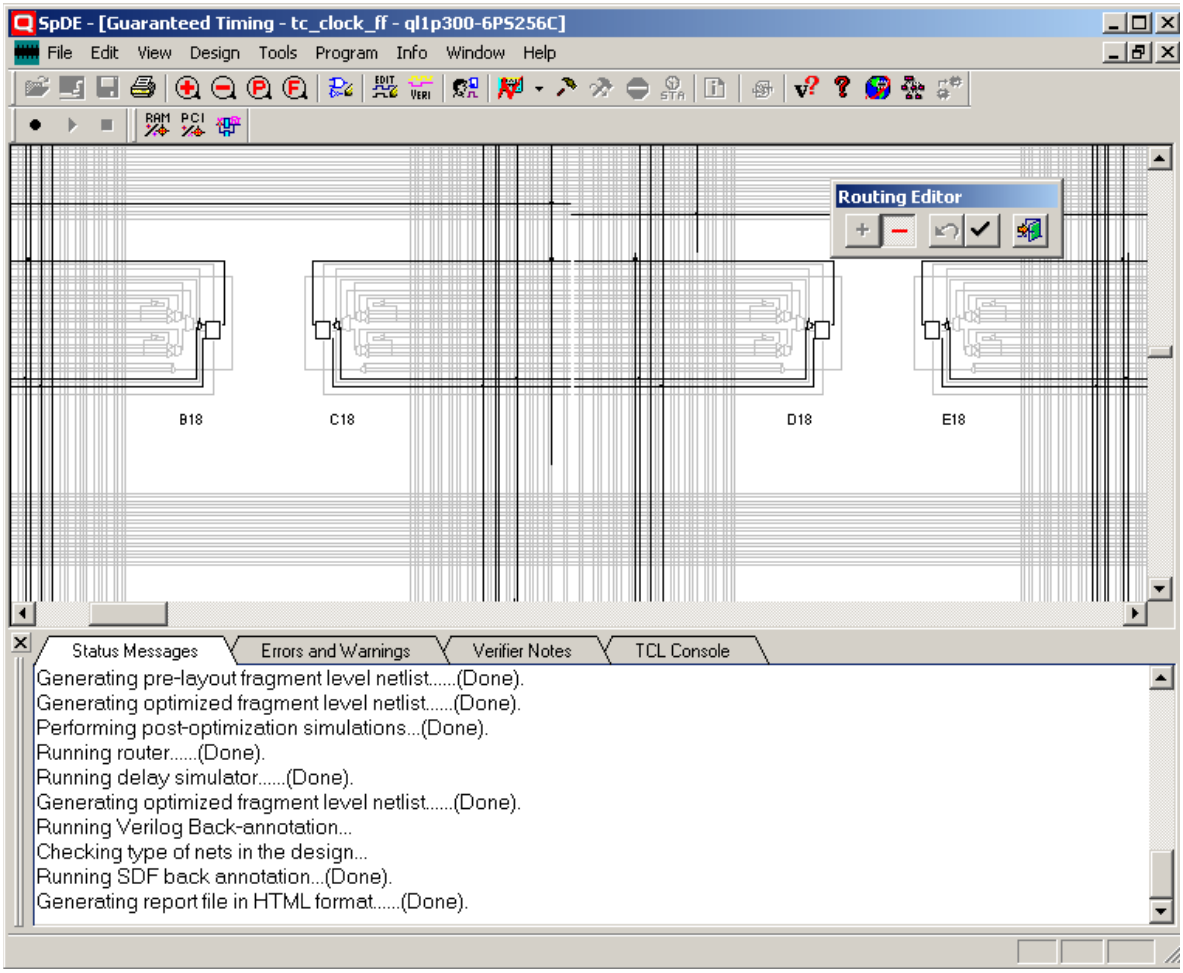
Table 14-1: Routing Editor Toolbar Commands

Icon	Function	Description
	Delete Route	Click on an existing route to delete a specific route on a net. If no net is selected, the route you clicked on selects that net. If you are in the process of editing a net (i.e., a net is already selected), you can only delete routes on that net, otherwise an error is reported.
	Add Route	Click on a wire with no existing route to add a route on that wire to the currently selected net. If no net is selected, this mode is disabled.
	Apply	Acts like the *Check* button; it specifies that you have finished hand-routing the selected net and would like to commit the changes. If the net not completely formed, you will not be allowed to commit the changes.
	Undo	Acts like the *Undo* button; it reverses back the editing step you performed.
	Exit	Acts like the *Exit* button; it exits the hand-routing mode and closes the Routing Editor.

© 2008 QuickLogic Corporation

14.6.3 Deleting a Route On a Net

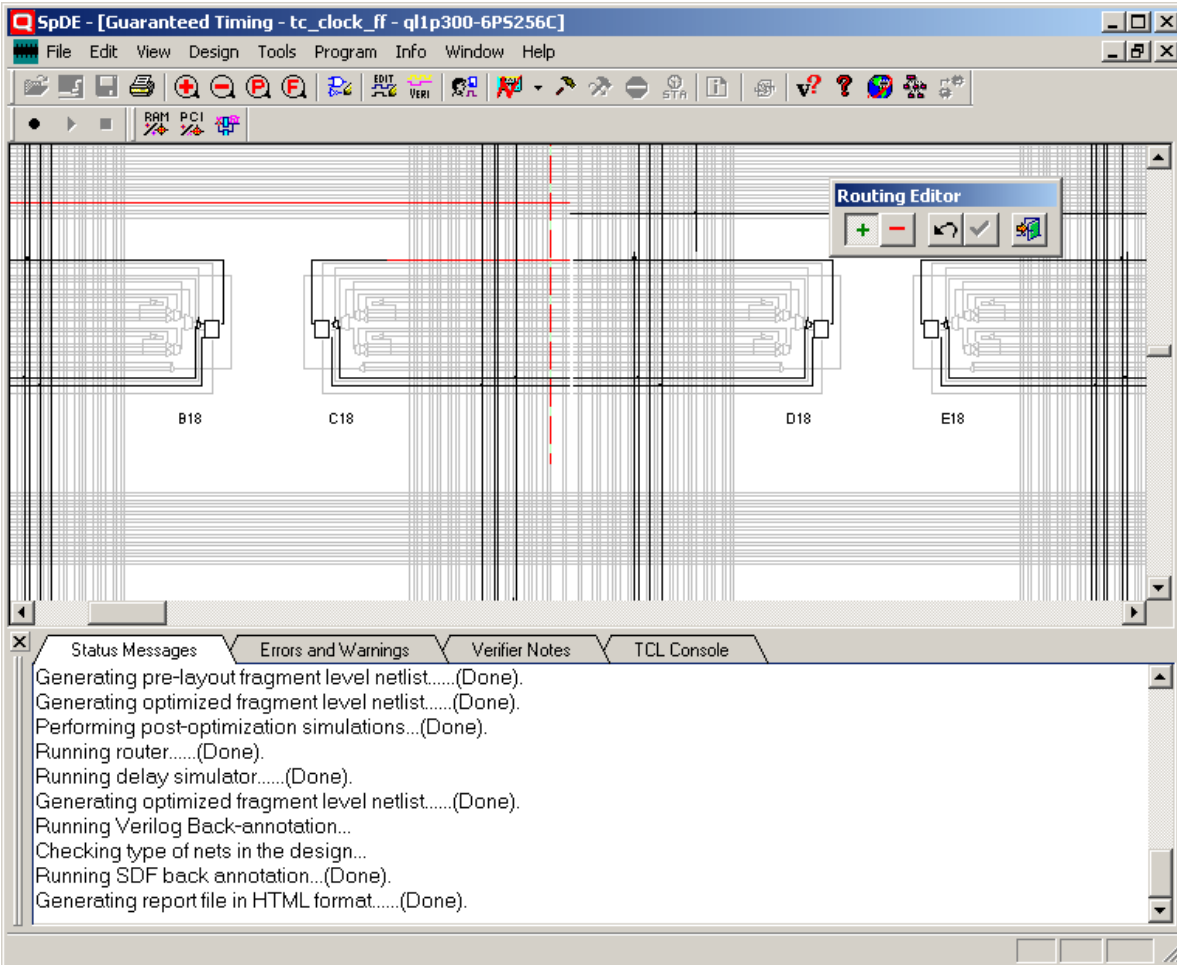
To delete a route on a net, select the net that you want to re-route, click on the **Delete Route** button in the Routing Editor, and then select the route that you want to delete. The selected route is deleted and the remaining part of the partially routed net turns red so that it can be easily identified from other nets.



© 2008 QuickLogic Corporation

14.6.4 Adding a Route On a Net

To add a route on a net, click on the **Add Route** button in the Routing Editor and select the route you want to add. The selected route is automatically added. If the net is completely routed, the recently added route along with the full net turns green. Otherwise, added routes would be displayed in red.



Whenever you select a valid route or wire, the cursor changes to "**+**" making it easier to make the selection. If you attempt an invalid selection, the cursor shape changes to "**x**".

NOTE: When selecting an object, watch the status bar for more information.

NOTE: Deleting a route on a net is also a way to select the net for editing. If you wish to switch to editing another net, you have to complete the currently selected net first.

Once a net is selected, it changes color to:

- Green—if the net in its current state is completely routed.
- Red—if the net in its current state is not complete.

© 2008 QuickLogic Corporation

14.6.5 Error Messages

Table 14-2 lists the Routing Editor error messages.

Table 14-2: Routing Editor Warning Messages

WARNING: Select **Delete Route** mode to start editing a net
You are clicking around without being able to select a net to edit. Nets can only be selected in Delete Route mode.
WARNING: Cannot delete route: Route on wire <wirename> does not belong to net being edited (<netname>)
You tried to delete a route that does not belong to the net you are currently editing.
WARNING: Cannot delete route: No route on wire <wirename>. Select route on net <netname> to delete
You tried to delete a route that does not exist, probably an unused wire segment, or you tried to delete a route that does not belong to the net you are currently editing.
WARNING: Cannot delete route: Non-signal net on wire <wirename>. Select a signal net route to delete
You tried to delete a route on a tie-hi/lo line. These nets cannot be edited.
WARNING: No route found on wire <wirename>: Click on existing route to delete.
You tried to delete a route that does not exist, probably an unused wire segment.
WARNING: Cannot delete route on wire <wirename>: Wire is connected to port
You tried to delete a route that attached to a port. This operation is not allowed as it can change the functionality of the circuit.
WARNING: Cannot delete route on wire <wirename>: Wire is tie-hi/lo wire
You tried to delete a route on a tie-hi/lo line. These nets cannot be edited.
WARNING: Cannot delete route on wire <wirename>: Deleting would cause an island route
You tried to delete a route on the currently edited net that could cause the net to split into three pieces. The routing editor allows only one disconnect in the net at any time.
WARNING: Cannot route on wire <wirename>: Wire is used by Net <netname>
You tried to add a route to a net on a wire where there already exists a route belonging to another net.
WARNING: Cannot route on wire <wirename>: Wire is connected to port <portname>
You tried to add a route to a net on a wire that attaches to a port. That could change the functionality of the design.
WARNING: Cannot route on wire <wirename>: Wire cannot connect to net <netname>
You tried to add a route to a net on a wire that cannot possibly connect to the net because this wire has no links to other non-port wires. This usually happens if you choose a clock wire that does not have links to many other wires.
WARNING: Cannot route on wire <wirename>: Doing so will cause a logic jog
You tried to add a route to a net on a wire that would cause a jog in the logic switch box. This is not permitted for programming reasons.
WARNING: Cannot route on wire <wirename>: Doing so will cause loop
You tried to add a route to a net on a wire that would cause a *loop* or a *near-loop* (see **“Manual Routing Editor” on page 163** for definition).
WARNING: Cannot route on wire <wirename>: Antenna detected on completed net

© 2008 QuickLogic Corporation

Table 14-2: Routing Editor Warning Messages (Continued)

You tried to add a route that would complete the net but may leave an *antenna* (a hanging route connected to the net but not going anywhere). This may not be obvious since the antenna could be far away from the route you are currently adding. Antenna checking is only done when the net is functionally complete and you would have to go back to where the antenna is, delete it and then add your chosen route.
WARNING: *** Route on wire <wirename> deleted for net <netname> ***
This is not an error but an information message.
WARNING: *** Route added on wire <wirename> for net <netname> ***
This is not an error but an information message.

14.6.6 Miscellaneous Issues

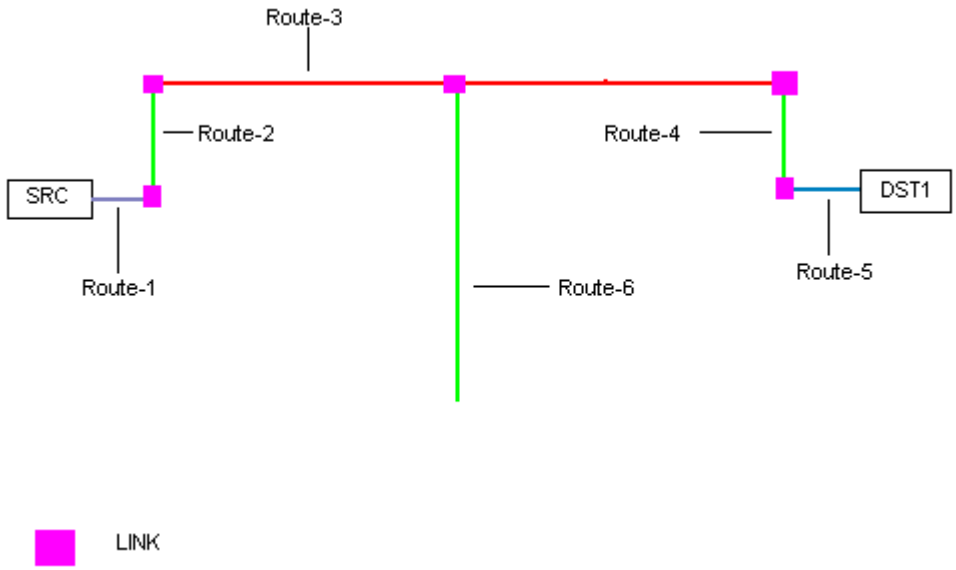
14.6.6.1 Detecting Whether Net Has Antenna

An antenna is detected on a completely routed net. If a net is complete (i.e., all of the destinations are connected), all of the routes, except those connecting to ports, will have two or more than two links (or connections). Routes connected with ports will have only one link. If there is any antenna, there will be a route with one (i.e., connected with some route) or no link (i.e., floating in between – not permitted) on it.

14.6.6.1.1 Condition for Antenna

For a route to act as an antenna, the number of links on it must be less or equal to '1', as shown in **Figure 14-4**.

Figure 14-4: Route Acting as Antenna



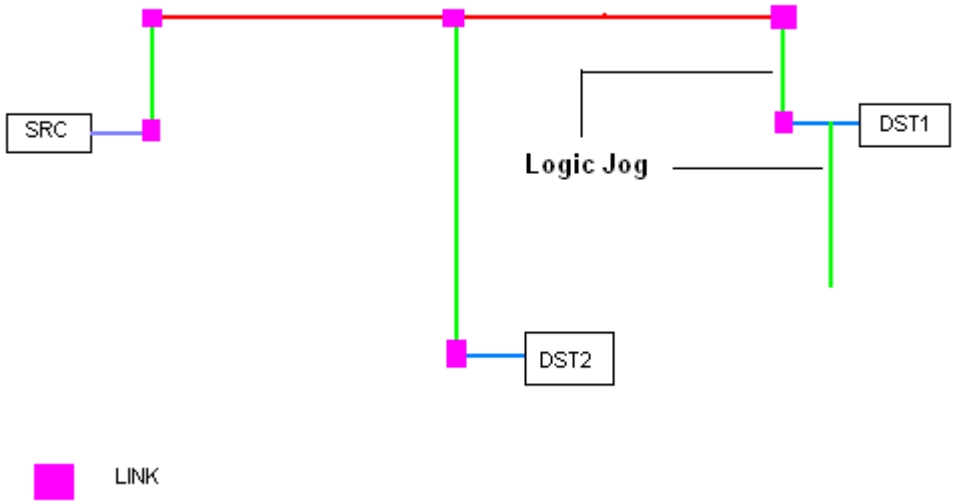
NOTE: In **Figure 14-4**, for Route-6 there is only one link, and therefore it act as an antenna.

© 2008 QuickLogic Corporation

14.6.6.2 Logic Jog

All of the routes coming out of a port or going into one are of a special type – they can have only one connection with the available routing resources. If such routes have more then one connection, then there is a *logic jog* and such a scenario should be avoided. An example of a *logic jog* is shown in **Figure 14-5**.

Figure 14-5: Logic Jog



© 2008 QuickLogic Corporation

Chapter 15
Back Annotation and Sequencer

This chapter addresses the following subjects:

- **“Functional Overview” on page 171**
- **“Delay Modeler” on page 172**
- **“Custom Temperature and Voltage” on page 174**
- **“Back Annotation” on page 174**
- **“Sequencer” on page 177**

15.1 Functional Overview

The Delay Modeler tool calculates the specific timing delays for the placed and routed pASIC device. The Back Annotation tool creates output files for logic simulation and fixing logic placement in QuickWorks. The Back Annotation tool puts the timing delays from the Delay Modeler together with the structural design information into output files for the selected simulator. The selected simulator can be the Active HDL or one from many other third-party vendors. The supported third-party simulators are listed in **Table 15-1 on page 175**. The Back Annotation tool also can be used to create files to fix flip-flop and I/O pad placement. See **Chapter 13, “Placer” on page 143** for details. The Sequencer tool is used to create an efficient sequence of links data to program the QuickLogic device.

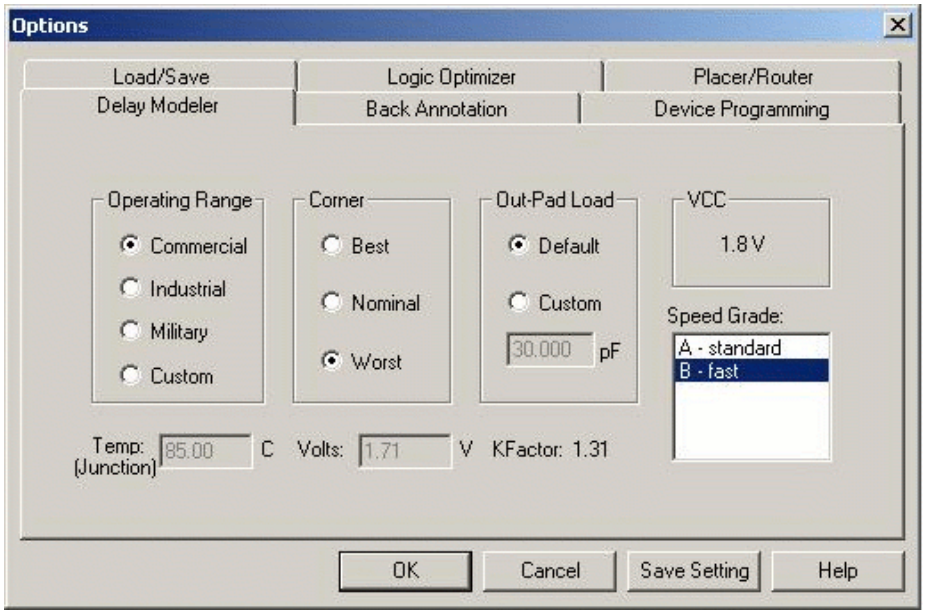
© 2008 QuickLogic Corporation

15.2 Delay Modeler

The Delay Modeler performs a comprehensive timing analysis, accounting for load, slew rate, signal propagation, and intrinsic delay. The tool uses a precise model of the pASIC device and calculates the effects of fanout, placement, and routing. The Delay Modeler can perform best-case, nominal, or worst-case analysis. The results of the worst-case analysis account for process variation, temperature, and voltage.

15.2.1 Delay Modeler Options Tab

To access the Delay Modeler options in SpDE, select **Tools>Options**.



15.2.1.1 Delay Modeler Operating Range

The **Operating Range** group box consists of the following options that control the temperature and voltage ranges used by the Delay Modeler:

- **Commercial**—default setting: temperature = 70 / 85 degrees Centigrade; voltage = 0.95 * operating voltage.
- **Industrial**—for industrial application operating conditions: temperature = 85 / 100 degrees Centigrade; voltage = 0.95 * operating voltage.
- **Military**—for military application operating conditions: temperature = 125 degrees Centigrade; voltage = 0.95 * operating voltage.
- **Custom**—allows user-specified temperature and voltage to be referenced.

NOTE: For PolarPro and PolarPro II devices, the *Custom* operating range selection is not available.

© 2008 QuickLogic Corporation

15.2.1.2 Delay Modeler Corner

The *Corner* group box consists of the following options that define the corners of the selected operating range:

- **Nominal**—default setting.
  For pASIC3, QuickRAM, and QuickPCI devices: 25 degrees Centigrade and 3.3 V regardless of the operating range selected.
  For Eclipse devices: 25 degrees Centigrade and 2.5 V regardless of the operating range selected.
- **Best**—selects the lowest temperature and highest voltage in the selected operating range.
- **Worst**—selects the highest temperature and lowest voltage in the operating range.

NOTE: Simulation should be performed at the *Worst* corner. For PolarPro and PolarPro II devices, the *Nominal* corner selection is not available (default corner is Worst).

15.2.1.3 Delay Modeler Out-Pad Load

The *Out-Pad Load* group box consists of the following options that select the capacitive loading on the output pins:

- **Default**: 30 pF
- **Custom**: allows a user-specified load in the range of 0 pF to 150 pF to be employed for all output pins.

15.2.1.4 Delay Modeler Speed Grade

The *Speed Grade* text box contains options to select the pASIC speed grade to be analyzed. The larger the number, the faster the device.

15.2.1.5 Delay Modeler Low Power

The *VCC* group box consists of two options that select the operating voltage. For PolarPro II devices, the user can select 1.5 V or 1.8 V. Devices that are 3.3 V-only (pASIC 3), 2.5 V-only (Eclipse), or 1.8 V-only (Eclipse II and PolarPro) do not have this option.

NOTE: The Delay Modeler has been tuned for peak accuracy within the recommended fanout ranges. High-Fanout nets that produce fanout warnings when the design is loaded into SpDE are calculated to the highest accuracy possible, but these results are not guaranteed.

© 2008 QuickLogic Corporation

15.3 Custom Temperature and Voltage

You can change **Temp** and **Volts** for each corner in the *Custom Operating Range* group box of the Delay Modeler. To change the temperature and voltage setting for the Custom operating range, edit the **Temp** and **Volts** fields in the Delay Modeler Options.

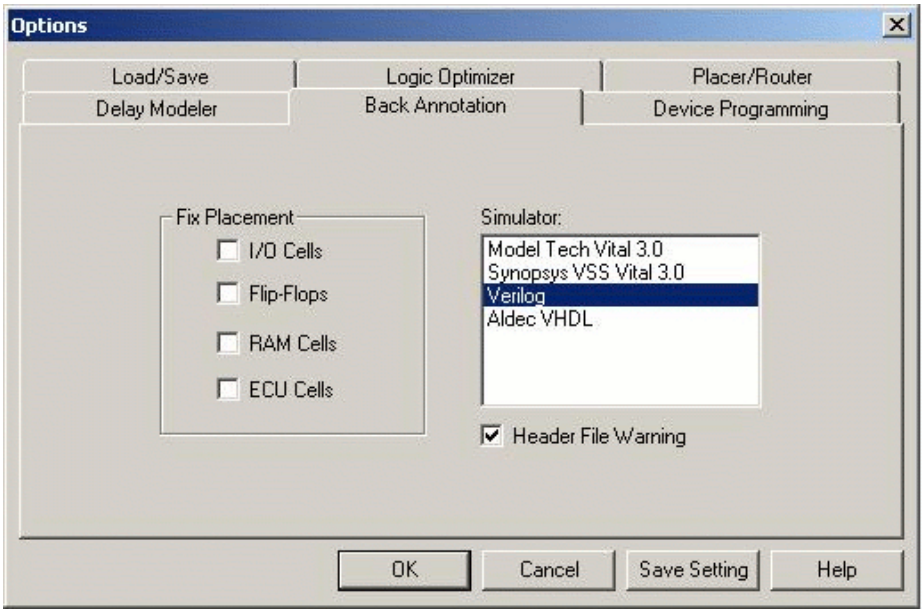
- Valid Temperatures are -55 to 125 degrees Celsius
- Valid Voltages are:
  - 3.0 V to 3.6 V for 3.3 V environments (pASIC3, QuickRAM, and QuickPCI devices)
  - 2.3 V to 2.7 V for 2.5 V environments (Eclipse and QuickMIPS devices)
  - 1.71 V to 1.89 V for 1.8 V environments (Eclipse II devices)

NOTE: The custom temperature and voltage setting is not available for PolarPro and PolarPro II devices.

15.4 Back Annotation

The Back Annotation tool produces files that send timing and placement information back to the design entry and simulation tools used. A variety of simulation tools are supported for back annotated simulation.

15.4.1 Back Annotation Options



© 2008 QuickLogic Corporation

A simulator can be chosen from the **Tools Options>Back Annotation** dialog box. This information will be automatically saved into the SPDE.INI file for subsequent sessions. By running the chosen simulator for any given design you create the simulated netlist. **Table 15-1 on page 175** indicates the files created by the Back Annotation tool depending on the Simulator setting.

Table 15-1: Simulation Back Annotation Files

Simulator	Design Kit	Files	File Function
Verilog	Verilog XL compatible	.vq .sdf	Verilog netlist delay back annotation file
Model Tech Vital 3.0	Model Technology's V-System VHDL Simulator	.vhq .sdf	Structural VHDL netlist Vital-3.0 compliant timing file
Synopsys VSS Vital 3.0	Synopsys VSS VHDL Simulator	.vhq .sdf	Structural VHDL netlist Vital-3.0 compliant timing file
ViewSim	Viewlogic	.vl .dtb	Intermediate file for spde2vl delay back annotation file
Vital-3.0 Compliant	Any Vital-Compliant VHDL Simulator	.vhq .sdf	Structural VHDL netlist Vital-3.0 compliant timing file
Active HDL	Active HDL Simulator	.vhq .sdf	Structural VHDL netlist Vital-3.0 compliant timing file
QuickSim	Mentor Graphics	.edo	EDIF file with timing
Intergraph EDIF	Intergraph	.edo .kf	EDIF file with timing delay scale reference file

15.4.1.1 Back Annotation for Verilog or VHDL Header Files

While running back-annotation for a Verilog (or VHDL) simulation, SPDE may issue a warning regarding a missing verilog (or VHDL) header file. If your design contains buses at the top level and you would like to preserve the interface to your top level verilog module (or VHDL entry), you must create a header file that contains ONLY the module (or entity) declaration. The header file should have a `.vh` (or `.vhh`) extension.

15.4.1.2 Verilog

The Verilog simulation netlist also requires a primitive file, which describes the functionality of the primitive components specified in the design `.vq` file. This primitive file is design-independent and is shipped with SpDE.

The primitive file is located in the `pasic\pasic_directory\spde\data\qlprim.v` directory. For PolarPro devices, `pasic\pasic_directory\spde\data\qlprimpp.v and` for PolarPro II devices, `pasic\pasic_directory\spde\PolarPro-II\qlprim.v.`

© 2008 QuickLogic Corporation

15.4.1.3 VHDL (Vital-Compliant, Synopsys VSS, Model Tech V-System, Active HDL)

The VHDL simulation netlist (Vital-Compliant, Synopsys VSS, Model Tech V-System, or Active HDL) also requires a primitive file, which describes the functionality of the primitive components specified in the `design.vhq` file. This primitive file is design-independent and is shipped with SpDE.

The VITAL 3.0 compliant primitive file is located in the `pasic\pasic_directory\spde\data\qlvt195.vhd` directory. For PolarPro devices, `pasic\pasic_directory\spde\data\qlvtlpp.vhd and` for PolarPro II devices, `pasic\pasic_directory\spde\PolarPro-II\qlvt1l.vhd.`

For tips in simulating with VHDL simulators, *see* **Appendix E, "Setting Up Third-Party Simulators" on page 547**.

15.4.1.4 Viewsim (ViewLogic)

When back annotating to *Viewsim*, the `SPDE2VL` program in the QuickLogic–ViewLogic Interface kit must be run after Back Annotation to create the ViewLogic `.wir` file needed for simulation. For more details, refer to **Appendix J, "Using ViewDraw with QuickWorks" on page 579**.

15.4.1.5 EDIF Simulation Netlists (LMC, QuickSim)

These netlists require simulation libraries for their respective simulation environment.

NOTE: The QuickLogic simulation library for LMC can be obtained through Synopsys; Intergraph's QuickLogic simulation library is available through Intergraph; the Mentor QuickSim QuickLogic simulation library can be obtained through QuickLogic; SusieCad's QuickLogic simulation library is supplied by Aldec.

© 2008 QuickLogic Corporation

15.5 Sequencer

The Sequencer tool is used to create an efficient sequence of links data to program the QuickLogic device. The efficiency of the Sequencer tool has been upgraded in V340HPC from 4-bit to 8-bit link sets for all QuickLogic Pinnacle family devices, the QL8325 device, and PolarPro devices. This upgrade improves programming time and productivity.

The new features in the V340HPC Sequencer tool are:

- 8-bit multibit Sequencer support for QuickLogic Pinnacle (0.25 µm) family devices, the QL8325 device, and PolarPro devices.
- 4-bit multibit Sequencer support for QL8150 and QL8050 devices.

15.5.1 8-Bit Multibit Sequencer Support for QuickLogic Pinnacle (0.25 µm) Family of Devices

The QuickLogic Pinnacle (0.25 µm) family devices, the QL8325 device, and PolarPro devices have eight voltage programming pins (VPPs) externally available for programming. Theoretically, eight links can be programmed simultaneously because each link in the 8-bit link set uses the unique VPP supply, available in all the above mentioned QuickLogic devices. The V340HPC Sequencer tool sequences up to 8-bit link sets, which can program up to eight links simultaneously.

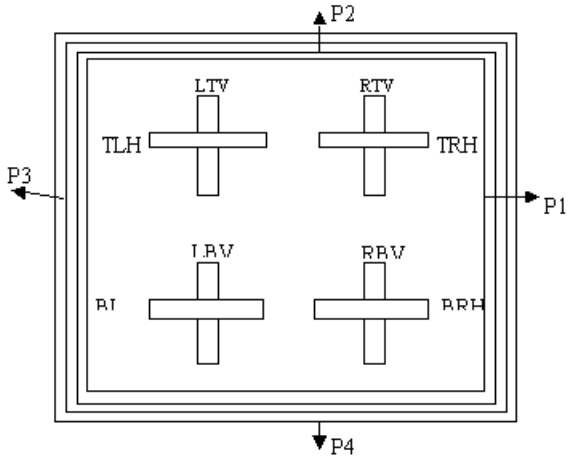
With this latest QuickWorks release, 8-bit multibit sequencing is done by default for the QuickLogic Pinnacle (0.25 µm) family devices, the QL8325 device, and PolarPro devices.

15.5.2 4-Bit Multibit Sequencer Support for QL8150 and QL8050 Devices

The QL8150 and QL8050 devices are from the 0.18 µm QuickLogic family of devices. These devices have a 4-VPP fully connected programming voltage topology. A maximum of four links can be programmed simultaneously using the QL8050 and QL8150 devices.

Figure 15-1 shows the QL8150 and QL8050 device programming voltage topology.

Figure 15-1: QL8150 and QL8050 Programming Voltage Topology



© 2008 QuickLogic Corporation

Chapter 16
Design Flows and Reference

This chapter addresses the following subjects:

- **“Functional Overview” on page 179**
- **“Design Flows” on page 180**
- **“SCS Tools Reference” on page 189**
- **“Active HDL Integration” on page 197**

16.1 Functional Overview

This chapter presents the design flows you may utilize as a designer: schematic-based (with or without Verilog and VHDL) or VHDL/Verilog-only. It also provides a general reference for the various tools and features introduced in the QuickWorks toolkit. For more specific details on the various tools and features, please refer to the *Synario Capture System User's Manuals*, the *Turbo Writer User’s Manual*, *Precision RTL Synthesis User’s Manual,* and Active HDL online documentation. Details on the SpDE logic optimization, placement and routing are available in other chapters of this manual.

The QuickWorks toolkit contains a suite of tools that is unprecedented in today’s market with prominent features such as:

- Enhanced Schematic Capture System
- Context Sensitive Editor with HDL Templates
- Verilog/VHDL Synthesis
- Mixed-Mode Design Entry
- Pre-layout and Post-layout Simulation

QuickWorks allows designers the flexibility, the power, and the ability to design in a variety of desired methods. With schematics, high level synthesis languages, or any combination of these design methods, designers can complete designs for all QuickLogic devices.

© 2008 QuickLogic Corporation

16.2 Design Flows

This section explains the different design flows available in QuickWorks such as traditional schematic design, detailing design flow, features, attributes and options that can enhance design performance and flexibility. Also covered are mixed schematic/HDL design flows, as well as Verilog-only and VHDL-only design flows.

16.2.1 Schematic-Based Design Flow

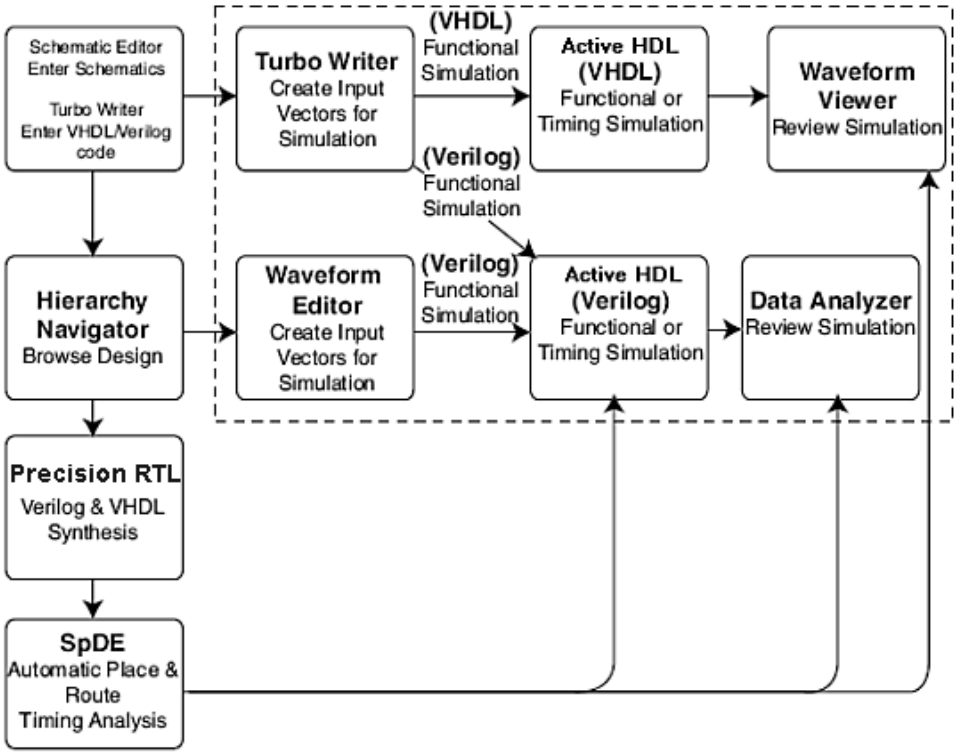
QuickLogic's QuickWorks design environment fully supports schematic-based designs (see **Figure 16-1**). A design can be entered as only schematics, or as a mixture of schematics, Verilog, or VHDL blocks.

Some types of logic, such as random glue logic or large structures such as adders or counters that exist in the schematic library, are suited to schematic entry.

Other types of logic are suited to textual description, such as decode logic, datapath logic, or state machines.

The QuickWorks toolkit integrates the SCS schematic environment tools with the Turbo Writer context sensitive editor and Precision RTL synthesis so that Verilog or VHDL blocks can be implemented as symbols in a schematic. In this way, a design can be entered using the most appropriate methods for each portion of the design.

Figure 16-1: Mixed Schematic-Based Design Flow



© 2008 QuickLogic Corporation

As shown in **Figure 16-1**, QuickWorks provides the capability to design in a mixed-mode environment. Designs can be captured in a schematic diagram with SCS schematic capture yet at the same time have unique symbols to reference VHDL and Verilog files. However, it is not possible to have both VHDL and Verilog blocks in the same design. These lower level HDL modules can be quickly described in the Turbo Writer editor which offers a host of features:

- Text editor emulation
- Context sensitive editing
- Verilog and VHDL language templates
- Syntax checking

Once a design is complete and checked using the DRC option, the HDL (Verilog or VHDL) netlist is generated and compiled by Precision RTL during the SpDE import of the design. When this design is loaded in SpDE, it can be placed and routed.

16.2.1.1 Entering a Design

A common design flow for a Schematic-based design consists of the following steps:

1. Use the SCS Schematic Editor to capture the schematic design:
   - Add symbols and wires
   - Add instance names to all symbols (optional)
   - Add net names to input and output nets
   - Add I/O markers to input and output nets
   - Run the schematic checker, fix any errors
   - Save the schematic
   - If this is not the top level schematic, create a symbol for the schematic (**File>Matching Symbol**)
2. Use Turbo Writer to enter any HDL files (Verilog/VHDL):
   - Make sure to use the correct file extension
     `VHDL:.vhd  Verilog:.v  QuickBoolean:.qeq`
   - Use Turbo Writer's Syntax Check for HDL files
   - Use Schematic Editor's **Add>New Symbol Block** to add HDL blocks to the schematic
3. Repeat steps 1-2 until the entire design hierarchy is entered.
4. Make sure to include I/O Pads on the top-level schematic.
5. Load the finished design into the SCS Hierarchy Navigator.
6. Export Verilog netlist.
7. Functionally simulate the exported Verilog netlist in Active HDL.
8. Repeat steps 1-7 until design is functionally correct.
9. Import the Verilog netlist into SpDE.
10. Run all Tools in SpDE including Back Annotation.
11. Simulate the post layout Verilog netlist (`.VQ`) created by SpDE.
12. Repeat Steps 1-11 until Post-Layout Simulation is correct.
13. Program a device in SpDE.

© 2008 QuickLogic Corporation

16.2.1.2 Adding Top Level I/O Pads to the Design

On the top-level schematic, you must include I/O pad symbols to represent the pins on the QuickLogic device. The pads to use depend on the pin characteristics you desire (consult **Table 16-1 on page 182**). Note that certain pin numbers are dedicated inputs that must be HDPADs (for pASIC3, QuickRAM, and QuickPCI devices), or HDPAD_25um's (for Eclipse devices) or CKPADs (for pASIC3, QuickRAM, QuickPCI, PolarPro, and PolarPro II devices), or CKPAD_25um's (for Eclipse devices). See the *Macro Library* chapter and the device pinouts for details. For pinout information, refer to the appropriate device datasheets available from the QuickLogic website [www.quicklogic.com].

Table 16-1: I/O Pad Symbols

Function	Pad Name (for QuickRAM, QuickPCI, PolarPro and PolarPro II)	Pad Name (for Eclipse and EclipsePlus)	Pin Type (from device pinout table)
Simple Input	INPAD	INPAD_25µm	I/O
Simple Output	OUTPAD	OUTPAD_25µm	I/O
Tri-State Output	TRIPAD	TRIPAD_25µm	I/O
Bi-directional Pin	BIPAD	BIPAD_25µm	I/O
High Fanout Input	HDPAD (not for PolarPro)	HDPAD_25µm	I
High Fanout Clock Input	CKPAD CKPAD2_DYN_EN (for PolarPro and PolarPro II)	CKPAD_25µm	I/CLK, GCLK/I, ACLK/I

Keep in mind that HDPADs/ HDPAD_25um's and CKPADs/ CKPAD_25um's have special drive characteristics, and must be used with care to avoid using up all express routing resources. For information regarding the rules concerning their use, refer to **"I/O Pads" on page 396**.

16.2.1.3 Adding Attributes to Symbols in the Design

Table 16-2 presents the four main attributes that can be added to symbols.

Table 16-2: Symbol Attributes

PLACE	Pin Location (I/O Pads) - example: 12 or 53 Logic Cell Location (Flip-Flops) - example: A2 or B5 Valid for I/O pad or flip-flop symbols only
FIXED	**Y** (or blank) - the PLACE attribute is valid **N** - the PLACE attribute is for viewing only Valid for I/O pad or flip-flop symbols only
PACK	**Y** - Level 0 Optimization will be performed on this symbol **N** (or blank) - symbol is treated normally Valid for any schematic symbol (meaningless for HDL blocks)
SYNMACRO	**Y** - symbol will *not* be run through Precision RTL synthesis **N** (or blank) - symbol will be treated normally Valid for any schematic symbol (meaningless for HDL blocks)

© 2008 QuickLogic Corporation

These attributes can be changed from the Schematic Editor (with the **Edit>Attribute>Symbol Attribute** command), or the Hierarchy Navigator (with the **Edit>Attribute>Symbol Attribute** command). The **Edit>Attribute>Symbol Attribute** command will open the Symbol Attribute Editor dialog box.

To select the symbol that you want these attributes to be changed to:

1. Click on the symbol in the schematic design and the click on the List All Attributes check box. A list of attributes with equal sign (=) will be displayed in the right window.
2. Select the attribute(s) to be changed and enter value for the attribute in the top window with the name of the attribute on the left side of the window. Attributes placed in the Hierarchy Navigator take precedence over the Schematic Editor.
3. If attributes are placed in the Hierarchy Navigator, then a **File>Save** command should be performed in the Hierarchy Navigator to create the `.tre` file which stores these attributes.

16.2.1.3.1 Setting PLACE and FIXED Attributes

In the Hierarchy Navigator, select **File>Back Annotate** to assign the PLACE and FIXED attributes to all pads and flip-flops in the design. These assignments will be based on the results from the most recent place and route, and whether the fixed I/O and Flip-Flop flags were set in the back annotation options in SpDE (for more information see **Chapter 15, "Back Annotation and Sequencer" on page 171**.)

16.2.1.4 Pin Placements for Bussed Pads (IPAD4, OPAD8, etc.)

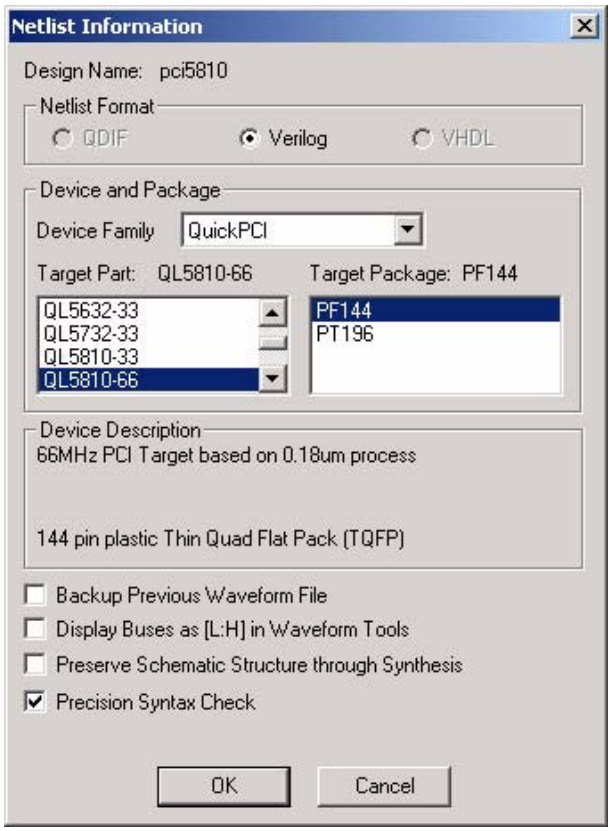
For bussed pads, such as `IPAD8` or `OPAD16`, it is not possible to set the `PLACE` attribute in the Schematic Editor. You must process to the step in the design flow where the design is loaded into the Hierarchy Navigator.

To push into the bussed pad macro select **View>Push/Pop**. You will see the individual I/O pads which you can put PLACE attributes on using **Edit>Attribute>Symbol Attribute**. Be sure to perform a **File>Save** in the Hierarchy Navigator to save the place attributes to the `.TRE` file.

© 2008 QuickLogic Corporation

16.2.1.5 Creating a Verilog, VHDL, QDIF or EDIF Netlist from the Hierarchy Navigator

In Step 5, there are multiple options for the type of Netlist you can create from the Hierarchy Navigator: a `QDIF` netlist (for non-PolarPro/PolarPro II devices), a `EDIF` netlist (for PolarPro and PolarPro II devices), a Verilog netlist, or a `VHDL` netlist. The following dialog box shows the Export QuickLogic Menu of Netlist types.



16.2.1.5.1 Verilog Netlist Format

This option is available if your design does not have any `VHDL` blocks.

If you have any Verilog in your design, you will be required to export a Verilog netlist. The resulting netlist can be imported into SpDE by selecting **File>Import Verilog** from SpDE. You will need to create a Verilog netlist if you wish to Simulate the design without timing delays.

When the Verilog netlist format is selected, a syntax check is automatically performed by the Precision RTL synthesis tool. This syntax check can be disabled by clicking the Syntax check box in the lower right of the Export QuickLogic window.

You can also choose to have the tool preserve your schematic structure, which has the benefit of keeping the VHDL netlist exactly like your schematic. The disadvantage is that logic optimization cannot be performed on the design by the synthesis tool when this option is selected.

© 2008 QuickLogic Corporation

16.2.1.5.2 VHDL Netlist Format

If you have any VHDL code in your design, you will need to export a VHDL netlist. The resulting netlist can be imported into SpDE by selecting **File>Import VHDL**.

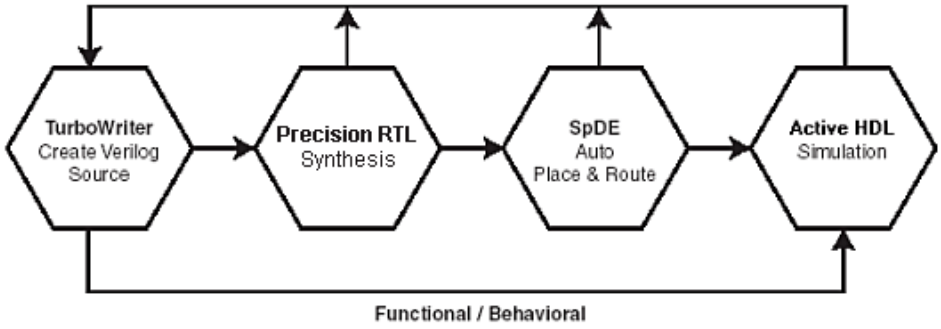
For schematic-only or mixed schematic/VHDL designs, you will need to create a VHDL netlist if you want to simulate the design without timing delays using a third party VHDL simulator. If you use QuickWorks Verilog/VHDL simulation, you will need to generate a pre-layout Verilog netlist.

When the VHDL netlist format is selected, a syntax check is automatically performed by the Precision RTL synthesis tool. This syntax check can be disabled by selecting the **Syntax** check box in the lower right of the Export QuickLogic window. You can also choose to have the tool preserve your schematic structure, which has the benefit of keeping the VHDL netlist exactly like your schematic. The disadvantage is that logic optimization can not be performed on the design by the synthesis tool when this option is selected.

16.2.2 Verilog-Only Design Flow

With the evolution of today's design methodology, synthesis is becoming ever more important. The QuickWorks toolkit contains a highly integrated text editor with editor emulation abilities, context sensitive Verilog templates, color coding, syntax checking, and a Verilog compiler (synthesizer). With such tools, the designer can describe a complete design at various levels of abstraction—from structural to behavioral.

Figure 16-2: Verilog Only Design Flow



As shown in **Figure 16-2 on page 185**, QuickWorks provides a quick and direct path for the design engineer. Designs can be quickly described in the Turbo Writer editor, compiled by Precision RTL during the SpDE import of the design, and placed and routed by the SpDE tools.

16.2.2.1 Entering a Design

A common design flow for Verilog-only design is:

1. Open a **New** file in Turbo Writer (specify the filename with `.v` extension).
2. Generate Verilog source code.
3. Perform a syntax check.
4. Repeat steps 1-3 until syntax errors are fixed.
5. Save the completed Verilog file.
6. Create a Verilog test-fixture by hand or with the graphical Waveform Editor.

7. Functionally simulate the Verilog design in Active HDL.
8. Repeat steps 1-7 until the Verilog design is functionally correct.
9. Import Verilog file to SpDE.
10. In the Precision RTL Setup Design option, select device and package before compiling.
11. Run all tools including Back Annotation.
12. Simulate the post-layout Verilog netlist (`.VQ`) in Active HDL.
13. Repeat steps 2-12 until the placed and routed design simulates properly.
14. Program a device.

16.2.2.2 Using the Turbo Writer Editor

Though any text editor can be used to describe the design, the Turbo Writer editor offers a host of very special features. The Turbo Writer allows the user to emulate from a variety of different standard text editing platforms such as - `vi, brief, epsilon` -. Context sensitive editing and Verilog/VHDL language templates provide shorthand entry for commonly used descriptions generating necessary keywords, brackets, and syntax. Color coded text allows for easy reading and quick comprehension. Syntax checking along with source debugging guarantees verification of the design.

16.2.2.3 Fixing Pins

In most cases it is very helpful and necessary to be able to fix pin locations and make special assignments. Create a complete design description in Verilog in a separate file with the `.sc` extension, rather then in the Verilog source file itself. This can be done with the following syntax:

```
#---Fixed I/O cells---
portprop ld ql_placement = “io15”;
portprop clk ql_placement = “io17”, ql_padtype=“CLOCK”;
portprop d[1] ql_placement = “io22”;
portprop d[0] ql_placement = “io12”;
portprop a[5] ql_placement = “H16”;
```

Notice that pin numbers must be prefixed with `io` (except in the case of grid-array package). Also notice that for signal name `clk` in the third line above, a `ql_padtype` property was added to indicate that a `CLOCK` pad should be used to access one of QuickLogic's two internal clock networks. The legal `ql_padtypes` are listed in **Table 16-3**.

Table 16-3: Legal ql_padtypes

INPUT	Use this for HDPADs (pins marked with I on the pinout table)
CLOCK	Use this for CKPADs (pins marked with CLK, GCLK/I, or ACLK/I)
BIDIR	Normal I/O pad (pins marked with I/O)

16.2.2.4 Compiling, Placing, and Routing the Verilog Design

Once your design is complete and correct, you can import your file into SpDE by selecting **File>Import Verilog**. A device, package, and any directives can then be selected and a compiled Verilog netlist generated.

© 2008 QuickLogic Corporation

After Precision RTL completes the compilation and synthesis, the netlist is automatically loaded into SpDE, where the design is automatically optimized, placed, and routed.

The Delay Modeler generates precise post-layout delays, which is back-annotated for simulation. Active HDL can be used to simulate the design, producing timing-accurate results. If desired, a functional simulation can be performed by simply entering the simulation stage after completing your Verilog description before place and route occurs. Finally, the device can be programmed for final test on the board.

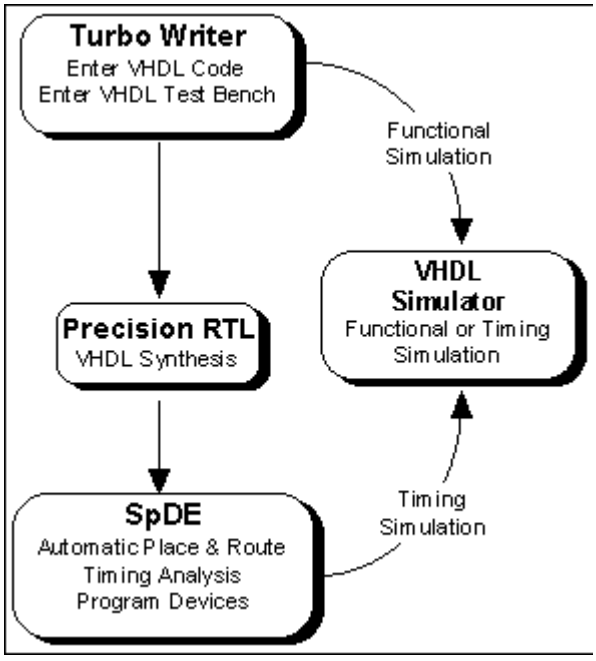
16.2.3 VHDL-Only Design Flow

For your VHDL design entry, QuickWorks includes Turbo Writer, a highly integrated HDL text editor that has VHDL and Verilog language-specific templates, color coding, syntax checking, and text editor emulation.

For VHDL synthesis, Precision RTL provides an easy-to-use interface, fast run time, and high quality synthesis results. Active HDL Simulator is included with QuickWorks to verify your design. QuickWorks also supports popular VHDL simulators such as Model Tech V Systems, Synopsys VSS, and other VITAL VHDL compliant simulators by exporting a VITAL VHDL netlist. Depending on the simulator you are using, you can follow one of the two design flows available for completing a VHDL design.

16.2.3.1 VHDL Design Flow With Active HDL Simulator or a Third Party VHDL Simulator

Figure 16-3: Recommended VHDL-Only Design Flow



1. Launch Turbo Writer and open a new file. Specify the design name with `.vhd` extension.
2. Enter your VHDL source code and perform a syntax check.

3. Correct your code until it is free of syntax errors.
4. Save your completed code.
5. Create a VHDL test bench for your design.
6. Perform functional/behavioral simulation using your selected VHDL simulator.
7. Import the VHDL netlist into SpDE.
8. In the Precision RTL Setup Design option, choose your device and package and then compile and synthesize. Click the **View Log** button if there are any errors.
9. In SpDE, select your simulator.
10. Run all tools.
11. Use Static Timing Analyzer to analyze timing.
12. Transfer the `design.vhq` and `design.sdf` files to your simulation environment and perform VITAL-VHDL timing simulation.
13. Repeat steps 2-13 to make changes to your design as needed.
14. Program a device.

© 2008 QuickLogic Corporation

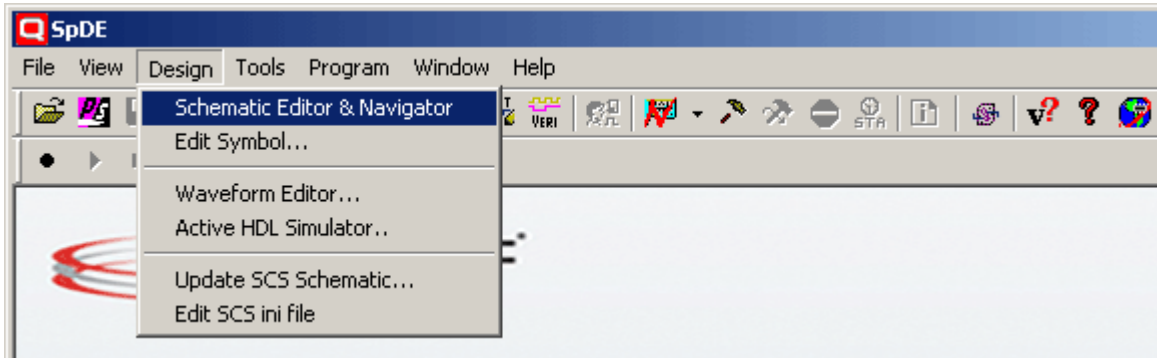
16.3 SCS Tools Reference

The Synario Capture System (SCS) provides schematic entry for QuickWorks. Other Hardware Description Languages (HDL) and boolean equation design entry tools are also available from third-party developers. This chapter provides a brief description of the SCS tools. The SCS Design Tools are documented in more detail in the *Synario Capture System User's Manual*.

To launch the **SCS Executive** double-click its icon . The SCS Executive may be used to access all SCS design entry tools: Schematic Editor, Symbol Editor, Waveform Editor.

To launch SpDE, double-click its icon . The **SpDE - Seamless pASIC Design Environment** is the framework by which all design flows are managed. All of the SCS tools can be launched from SpDE's **Design** menu.

While it is always possible to access the SCS design tools from the SCS Executive, SpDE provides streamlined access to the SCS tools, via the Design menu, which includes Design Entry, Simulation, and Project Configuration.



16.3.1 Schematic Editor & Navigator

To launch the Schematic Editor, in SpDE, select **Design>Schematic Editor & Navigator**. The Schematic Editor is used to create each schematic in the design. The Schematic Editor & Navigator option always invokes the Hierarchy Navigator tool first. To open the schematic design, use the **Edit>Edit Schematic** option. Although multi-sheet schematics can be created, this is generally discouraged as multi-sheet schematics indicate a flat design style rather than a hierarchical design style. It is much easier to navigate a fully hierarchical design than a flat design that includes many multi-sheet schematics.

A common design flow within the Schematic Editor is as follows:

1. Select the device and package, if the Editor is opened to create a schematic.
2. Add symbols and wires.
3. Add instance names to all symbols (optional).
4. Add net names to input and output nets.
5. Add I/O markers to input and output nets.
6. Run the schematic checker.
7. Fix any errors (re-run steps 1-6).
8. Save the schematic.
9. If the schematic is not the top level, create a symbol for the schematic.

© 2008 QuickLogic Corporation

NOTE: Power and Ground Connections are made in the Schematic Editor by naming the net. To tie a net high, add a Net name of VCC. To tie a net low, add a Net name of GND. If the net name is attached to the end of a vertical wire, the text of the name will be replaced with a snazzy VCC or GND symbol.

NOTE: Net names and instance names should include only alphanumeric characters or underscores (_). Some non-alphanumeric characters (asterisk, slash, dot, dash) are known to cause problems in the Waveform Editor and the simulator.

16.3.1.1 Creating Buses in the Schematic Editor

Busses are drawn in the schematic editor by selecting **Add>Wire**. The wire is converted to a bus by giving the wire a bus net name. Add a Net name such as `FOO[7:0]` for an ordered bus, or `A,B,C,D` for a simple bus.

16.3.1.2 Grounding a Bus

A bus can be tied to a constant value in the Schematic Editor by naming the bus. To tie a 4-bit bus low, Add a Net name of `GND[3:0]`. If the lower 4 bits of an 8 bit pin named `D[7:0]` need to be grounded, and you wish to attach the upper 4 bits to a bus named `A[3:0]`, then add a bus to the pin named `A[3:0],GND[3:0]`.

16.3.1.3 Connecting Buses

In the SCS Schematic Editor nets can be connected by net name alone. This is also true of busses. If you use a bus named `A[7:0]` in one part of the schematic, and `A[3:0]` in another part of the schematic, then these busses are connected *by their names alone*.

NOTE: The busses don't have to be physically touching for the connection to take place. In fact, it is illegal in SCS to physically connect two busses with different sizes or names.

16.3.1.4 Using Iterated Instances in the Schematic Editor

Another powerful feature in the Schematic Editor is the ability to handle iterated instances. Iterated instances allow a single instance of a symbol to represent many identical instances in a parallel connection.

This feature is useful when connecting an array of identical macros to a bus such as buffers, input pads, output pads, flip-flops, etc. For example, suppose you wanted to create a 32-bit input bus and attach it to 32 `INPAD`s. You can use an iterated instance of the `INPAD` symbol. Place the INPAD symbol on the schematic, and with the **Add>Instance Name** command the status line reads: Instance Name - Enter Name = . Enter IN[31:0] at the status line and press the Enter key. Place IN[31:0] at the INPAD symbol by clicking on the INPAD symbol. By adding this bus-type instance name, the one `INPAD` has been magically transformed into a 32-bit `INPAD`. You can now attach a 32-bit bus to the input and output of this Iterated Instance.

16.3.2 Symbol Editor

The Symbol Editor may be used to create intricate symbols for hierarchical elements.

© 2008 QuickLogic Corporation

To load the Symbol Editor, in SpDE select **Design>Edit Symbol**. Most of the symbols in the pASIC Macro Library, for example, were created using the Symbol Editor. Keep in mind, however, that the Schematic Editor has the ability to create a symbol for any schematic (using the **File>Matching Symbol** command). These automatically created symbols may be edited with the Symbol Editor if desired. The most common use for the Symbol Editor is to change a symbol type to `CELL` for automatically generated Master Cell symbols. Refer to the *Macro Library* chapter for more details on creating Master Cells.

16.3.3 Hierarchy Navigator

The Hierarchy Navigator is the heart of QuickWorks' schematic design environment.To launch the Hierarchy Navigator, in SpDE, select **Design>Schematic Editor & Navigator**. The Navigator builds an in-memory database of the completed design, and provides the other tools with an interface to this database. Within QuickWorks, the Navigator provides four important functions:

- Examines the complete design hierarchy (**View>Push/Pop**).
- Assigns attributes within the design hierarchy (**Edit>Attribute>Symbol Attribute**).
- Generates either a QDIF, EDIF, Verilog, or VHDL netlist (**Tools>Export QuickLogic**).
- Interactively simulate the design (**Simulation** menu).

The ability to examine the hierarchy is a convenient means of browsing the design in an interactive manner. While examining the design, attributes may be assigned for fixed placement of cells. It is important to note that these attributes are assigned hierarchically—each attribute is assigned to a unique instance of a component within the design. To add attributes, select **Edit>Attribute>Symbol Attribute**, then select the component instance. Once you add attributes in the Hierarchy Navigator it is important to select **File>Save** from the Navigator to add these attributes to the Navigator's attribute file (extension `.TRE`).

16.3.3.1 Tree (.TRE) and Schematic (.SCH) Files

From within SpDE, select **Design>Navigate Hierarchy**. A file selection dialog box appears. At first, the file selection box looks for `.tre` files, but if you clicked **NEW**, the file selection box looks for `.sch` files.

The Navigator can be opened with either the `.tre` or the `.sch` file.

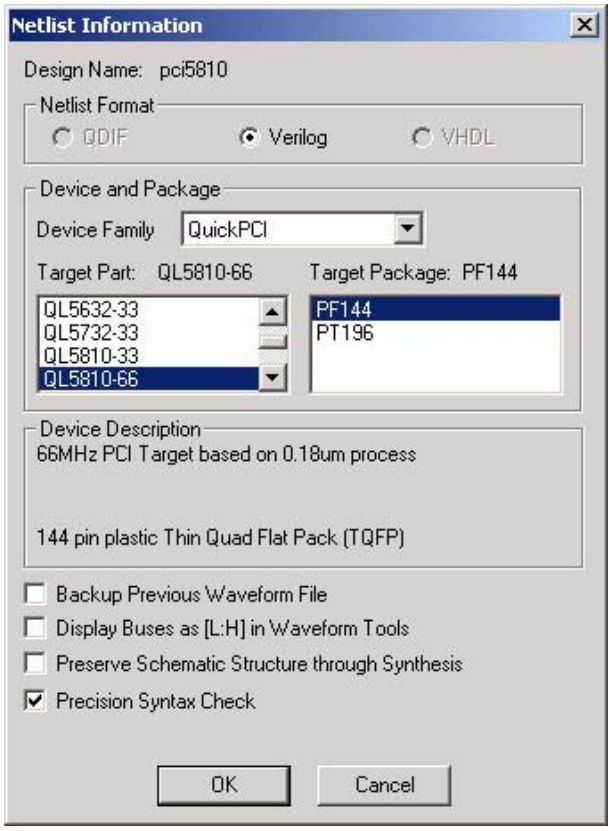
- The `.sch` file is the schematic file created by the Schematic Editor.
- The `.tre` file contains Navigator-only information, such as the last zoom level and window position, and what attributes have been added in the Hierarchy Navigator. The `.tre` file is created with the **File>Save** command in the Hierarchy Navigator.

NOTE: Occasionally when opening the Hierarchy Navigator, an **Invalid Instance Name** error will occur. This error is not as serious as it sounds. It means that the `.tre` file contains an attribute for a component in the design (an instance) which has been deleted or changed. Since it can no longer find the same instance, it reports the error: `invalid instance name`. Simply doing a **File>Save** from the Navigator will re-save the `.tre` file without this stray attribute, and the error will not re-occur.

© 2008 QuickLogic Corporation

16.3.3.2 Exporting the Netlist

The Hierarchy Navigator's Tools menu includes one item: **Tools>Export QuickLogic**. This is used to generate a netlist file used by SpDE. This menu item brings up a Netlist Information dialog box in order to specify the desired netlist, device part, and device package. This allows for quite a bit of flexibility to the designer. Once the netlist file is created, it can be loaded into SpDE from the **File>Import** command.



The QDIF (or EDIF) netlist can only be selected for designs that utilize the schematic design flow. If the QDIF/EDIF netlist is selected, the design is ready to be loaded into SpDE using the **File>Open** command. If the tree file (file extension `.tre` is saved from the Navigator after an Export QDIF/EDIF command is invoked, an attribute with the selected device will be stored in the `.tre` file. The selected device becomes the default in the Netlist Information dialog box.

On the other hand, Verilog or VHDL netlists can be selected for schematic, Verilog, VHDL, or combinations of these, except Verilog blocks cannot be exported to a VHDL netlist, and vice versa. However, if Verilog or VHDL is used in the design entry, the dialog box will automatically default to Verilog or VHDL, respectively. If Verilog is selected, the schematic portion of the design is written as a structural Verilog netlist (`design.v`). If `VHDL` is selected, the schematic portion of the design is written as a structural `VHDL` netlist (`design.vhd`). When a Verilog or VHDL netlist format is selected, a syntax check is automatically performed by the Precision RTL synthesis tool. To disable this syntax check, select the **Precision Syntax Check** check box in the Netlist Information dialog box. You can also choose to have the tool preserve your schematic structure, which has the benefit of keeping the VHDL netlist exactly like your schematic. The disadvantage is that logic optimization can not be performed on the

© 2008 QuickLogic Corporation

design by the synthesis tool when this option is selected. Some older designs may contain macros with busses in `(LSB:M5B)` format. To maintain backwards compatibility with these designs, you can select the Display Busses as `[L:H]` in Waveform Tools. Otherwise, the tools will recognize each bus as `(MSB:LSB)` format.

When selecting **File>Import>Verilog** or **File>Import>VHDL** from SpDE, Precision RTL is invoked and a complete Verilog or VHDL netlist is compiled, synthesized, and imported directly into SpDE. From here, the place and route tools can be run and the design can be mapped into the pASIC architecture.

16.3.4 Simulation

Simulation can be performed from the Hierarchy Navigator, from the Active HDL menu, or from the Design menu within SpDE.

16.3.5 Updating Schematics

In SpDE, the **Design>Update SCS Schematic** command verifies that schematics and symbols are up-to-date. You need to update your SCS Schematic after updating to a new SpDE version, or when a symbol has been edited.

16.3.6 Changing SCS Design Options

In SpDE, the **Design>Edit SCS ini file** command loads the SCS.INI Editor. The SCS.INI Editor provides easy access to all configuration options within the Synario Capture System. These include display colors, sheet sizes, printing and plotting options, and library directory configurations. After changing options, click **OK** in the SCS.INI Editor to save the options.

16.3.7 Turbo Writer

Turbo Writer is a fully configurable text editor with HDL templates, context sensitive editing, color coded presentation, and a host of other features and can be launched from the Design menu. Refer to the *Turbo Writer User's Manual* included with your QuickWorks package for more information.

NOTE: If you prefer another text editor, you can configure the menu item to launch any text editor you have on your system.

16.3.8 Verilog/VHDL Entry

The QuickWorks toolkit supports the incorporation of Verilog or VHDL code with schematic design entry. Precision RTL combined with the QuickLogic netlist generator provides a powerful synthesis engine and an efficient method to design in Verilog or VHDL. A complete design can now be described in either Verilog or VHDL. In addition, Verilog or VHDL blocks can be inserted into a larger schematic design,

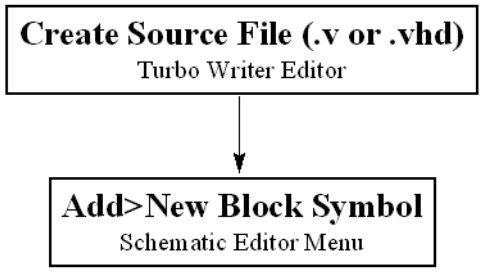
NOTE: You should not have both Verilog and VHDL blocks in the same design because Verilog and VHDL blocks cannot co-exist in the same schematic design.

© 2008 QuickLogic Corporation

Devices can be described at a very abstract and behavioral level or at a very detailed and structural level. In either case the synthesis engine provides excellent results in an incredibly short amount of time

The Verilog and VHDL source code can be entered by any text editor. However, QuickLogic provides the Turbo Writer—a robust text editor that is fully configurable with a host of special features. If Verilog is used only to describe a portion of a design, the Verilog or VHDL source code is represented in an SCS schematic by an SCS symbol (`file.sym`) which references a Verilog or VHDL source file (`file.v` or `.vhd`). **Figure 16-4** illustrates this flow.

Figure 16-4: Flow Diagram for Creating Verilog/VHDL Blocks



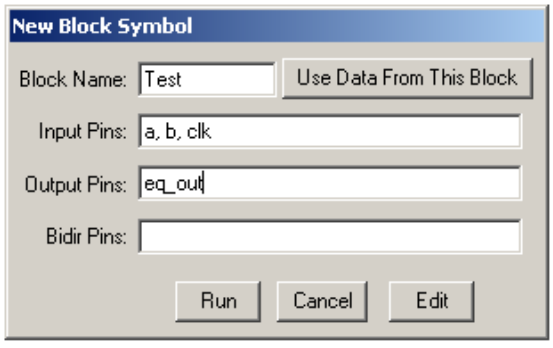
If the Verilog or VHDL code is used to describe a complete design (**Figure 16-4** shows the basic process for using Verilog or VHDL blocks in designs.), then the source file (`.v` or `.vhd`) can be imported directly into SpDE by selecting **File>Import Precision RTL**.

NOTE: It is important to remember that either Verilog or VHDL can be used to describe small pieces of functionality or entire designs.

16.3.8.1 Creating a Verilog or VHDL Symbol

Creating an SCS Symbol for a set of Verilog or VHDL files is a completely automatic process using the SCS automatic symbol generator located in the SCS Schematic Editor.

1. From the Schematic Editor select **Add>New Block Symbol**. The New Block Symbol dialog box opens.



The **Block Name** is the name with which you will call the Verilog symbol. It is also the name of the Verilog or `VHDL` source file (without the extension `-.v or.vhd`)

© 2008 QuickLogic Corporation

The **Use Data From This Block** button is not used for Verilog or VHDL block symbols. Notice that a Block Name of `Test` was used in the above example. This is the root name of the SCS symbol file that would be created (`test.sym`). No matter what case you use for the symbol name, the DOS filename is not case-sensitive. Therefore, the Verilog netlister assumes that the symbol has a lower-case name. The module name in the Verilog file must be lowercase. VHDL is not case-sensitive, it is therefore not necessary to have the case of the symbol name match the case of the entity name. However, you do need to make sure the source file name and the entity name are the same as the symbol name.

Be sure to type in the input pins, output pins, and bi-directional pins just as they will appear in the Verilog declarations. The above example specifies three input inputs : `a, b,` and `clk`. and only one output pin: `eq_out`.

NOTE: Since Verilog is case sensitive, be sure that you use the same case as in the Verilog source file.

NOTE: If you are including a bus as a pin, be sure to refer to the bus with = signs around the bus name, as in =A[7:0]=, when you type the bus name into the Create Symbol dialog box. Otherwise, the bus will be expanded into individual bits on the symbol, which will not connect properly to a bus port specified in the Verilog description. Also, be sure that the bus name on the symbol has the same bus width and bus order as in the Verilog or VHDL source file.

NOTE: When entering Input Pins, Output Pins, and Bi-directional Pins it is sometimes desired to use busses. Bus names (like a`[7:0]`, which would represent the signals a[7] through a[0] in a Verilog file or `a[7 downto 0]` equivalent in a VHDL file) have the syntax =`a[7:0]`= to identify them as a single bus pin. For example if you have input signals `c,d,a[7:0]` and e, in the **Input Pins** box you would enter `c,d,=a[7:0]=,e`.

2. Click **Run** to create and save the symbol file.

16.3.8.2 Creating Verilog (.v) and VHDL (.vhd) Source Files

This section describes how to enter Verilog or VHDL code into the QuickWorks environment. It does not discuss the Verilog or VHDL language itself—coding, syntax, keywords, etc. For more information on the languages, refer to Verilog or VHDL documentation.

The Verilog/VHDL source code can be entered using any text editor. However, QuickLogic provides *Turbo Writer* – a fully configurable text editor with many special features. In SpDE, select **Design>Text Editor** to open a text editor in order to edit or create Verilog files. While the Text Editor currently defaults to the Turbo Writer Text Editor, you may select any editor you desire. The text file you create should have the same root name as the symbol, with the extension `.v` for Verilog and `.vhd` for VHDL. as in `test.v` or `test.vhd` for example.

NOTE: An SCS symbol can be associated with a Verilog source file (`.v`), a VHDL Source file (`.vhd`), or an SCS Schematic (`.sch`). Be sure you don't have multiple source files with the same root name, or you may not get what you expect.

16.3.8.2.1 Launching Turbo Writer

Launch Turbo Writer by double-clicking its icon, or in SpDE, by selecting **Design>Text Editor**. This text editor may be used to describe Verilog or VHDL source files. Turbo Writer offers enhanced features for working with HDL files. Syntax checking, and test-bench generation are both available from the HDL menu in Turbo Writer.

© 2008 QuickLogic Corporation

16.3.8.2.2 Using Context Sensitive Editing

One of the many powerful features available in this editor is context sensitive editing. Templates for Verilog and VHD have been created based upon commonly used key words and phrases. These templates are based upon the extension of the file being edited - (`.v` for Verilog; `.vhd` for `VHDL`). This provides shortcuts to your coding and provides all the syntax associated with that keyword. By typing in the shortcut keyword followed by a `<tab>`, the full word or phrase with associated syntax is displayed. Refer to the Turbo Writer documentation for a list of supported templates and macros.

16.3.8.2.3 Understanding Color Coding

Another helpful feature is color coded text. This helps in the reading and understanding of your code. By default:

- All code defined by the text as a comment is colored in green (Verilog uses `// or /* */` for comment, and VHDL use `--`).
- All keywords are colored in blue.
- All numbers are colored in brown.

You may leave this color coding as it is or change the colors to something that is more pleasing to yourself.

16.3.8.2.4 Using the Syntax Checker

When editing a Verilog or VHDL file (extension - `.v` or `.vhd`) in the Turbo Writer editor, it is possible to perform a syntax check. Select **HDL>Synthesizer Syntax Check**, then wait a few seconds and read the status bar at the bottom of the TurboWriter dialog box. If any syntax errors are found, the error is shown in red, and the cursor moves to the line in the file containing the syntax violation. Use the **Next Error** button in the Turbo Writer toolbar to move to the next syntax error. To determine the function of the buttons in the Turbo Writer toolbar, simply place the mouse cursor over the button. A box appears adjacent to the mouse explaining the button's function.

NOTE: The Syntax Checker automatically saves your Verilog or VHDL file before performing a syntax check. Be sure you keep this in mind, so that you do not save unwanted changes.

© 2008 QuickLogic Corporation

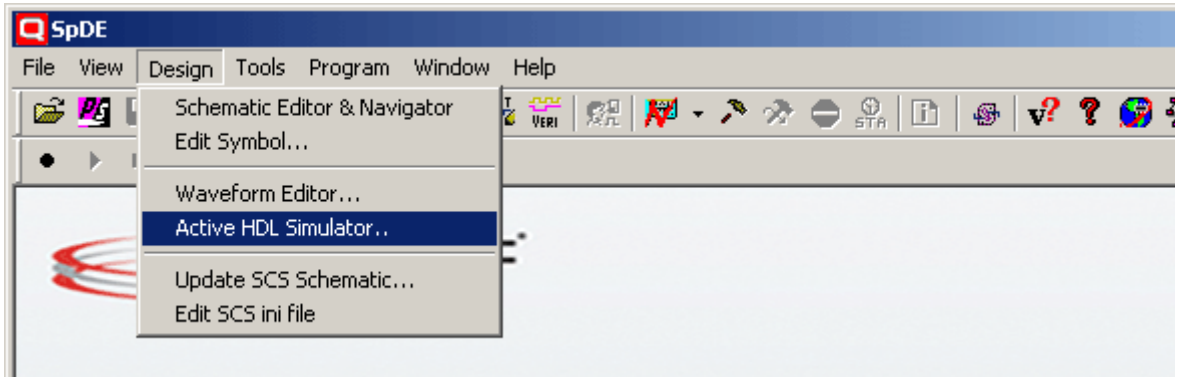
16.4 Active HDL Integration

The Active HDL Interface opens when invoking the Active HDL Simulator in SpDE. It accepts the simulation design information from the user, invokes the Active HDL Simulator, creates the workspace in the simulator, and compiles the design files in the simulator. To complete the simulation, the user only has to add waveforms and run simulation for the required time in Active HDL.

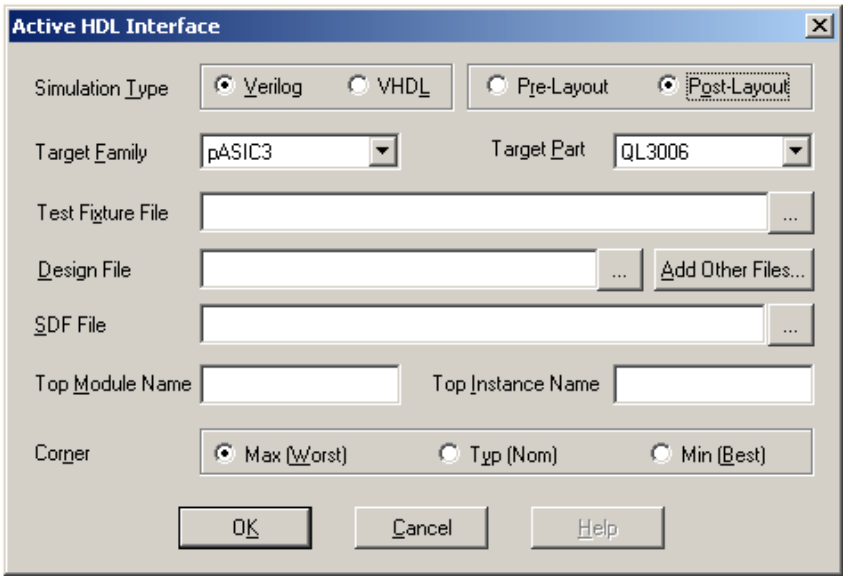
16.4.1 SpDE Menu Option

Active HDL Simulator can be invoked in SpDE by any of the following ways:

- Menu option—**Design > Active HDL Simulator**.
- Hot key—**Alt+D+H**
- Toolbar icon—**Active HDL Simulator**



16.4.2 Active HDL Interface Inputs

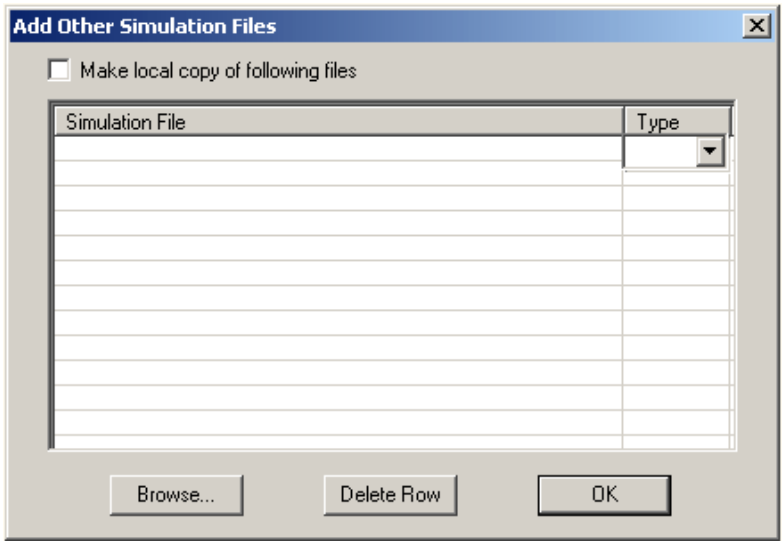


NOTE: Based on the simulation type (pre-layout or post-layout) selected, some fields are disabled. If the design is loaded, some information is automatically filled in.

- **Simulation Type**—Select one type from each group (Verilog or VHDL; Pre-Layout or Post-layout)
- **Target Family**—The family of the targeted part. This is required only to filter the part list, which makes part selection easier.
- **Target Part**—The part to which the HDL is to be targeted. For post-layout simulation, when the design is loaded, this field is disabled. The part of the chip file loaded is shown automatically.
- **Text Fixture File**—Text fixture file (`.tf, .tb,` etc.) with path:
  - Can be entered or browsed using the browse button. In the browse button, `*.tf`, `*.v` and `*.*` filters are shown for Verilog and `*.tb`, `*.vhd` and `*.*` are shown for VHDL.
  - In post-layout simulation, when a design is loaded in SpDE, if a test fixture file with the design name exists in the design directory, it will be displayed.
- **Design File**—The top-level Verilog or VHDL file for the simulation. In post-layout simulation, when a design is loaded in SpDE, if a `.vq` or `.vhq` file with the design name exists in the design directory, it will be displayed. If both `.vq` and `.vhq` files are present, the `.vq` file will be displayed.
- **Add Other Files**—Click this button to invoke a dialog box where you can add or remove files. If there are any standard files required for a part, they will be added automatically.
- **SDF File**—The SDF file for post-layout simulation. In post-layout simulation, when a design is loaded in SpDE, if a `.sdf` file with the design name exists in the design directory, it will be displayed. Otherwise, the information has to be entered manually. This field is disabled for pre-layout simulation.
- **Top Module Name**—The top module name for the simulation.
- **Top Instance Name**—The instance of the top module. It is required for post-layout simulation, while adding a `.sdf` file to the simulator. This field is disabled for pre-layout simulation.
- **Corner**—It is required for post-layout simulation, while adding a `.sdf` file to the simulator. This field is disabled for pre-layout simulation.
- Click **OK** to invoke the Active HDL Simulator. All simulation files are added into a workspace. The simulation files are compiled automatically.
- Click **Cancel** to cancel the operation of invoking the Active HDL Simulator.

© 2008 QuickLogic Corporation

The Add Other Simulation Files dialog box opens when the **Add Other Files** button is clicked. If there are any files specific to the selected part, those are shown automatically. For example, if the part is QL5020-33, the `pcit32n_design.v` file is displayed. Users can add more files required for the simulation.



- **Make local copy of following files**—Selecting this check box creates local copies in the simulation directory of all added files.
- **Simulation File**—Enter the path and file name of the simulation file.
- **Type**—Select the file type (auto, verilog, vhdl, txt, wave, symbol, or tcl); default is *auto*.
- Click **Browse** to select a file after selecting the row into which the selected file should be copied.
- Click **Delete Row** to delete the selected row.
- Click **OK** to save the information and close the dialog box.

When you click **OK**, an Active HDL macro script is created in the design directory with the name `<design name>_<pre or post>.do` (e.g., `counter_pre.do`). The latest version of the Active HDL Simulator available on the machine is detected and invoked and the macro script is executed in the simulator. During the execution of the macro, an Active HDL design file is created, opened, and all simulation files are added to the design workspace in Active HDL. The simulation files are then compiled and the simulation is initialized (set top module). The required waveforms have to be added in Active HDL before running the simulation.

NOTE: For a detailed information about the functions in Active HDL, refer to relevant Active HDL user manuals.

For more information, see also:

- *QuickWorks Tutorial User Manual – Sections: Pre-Layout Simulation Using Active-HDL* and *Post-Layout Simulation Using Active-HDL*

© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation

Chapter 17
Hard Macros

This chapter addresses the following subjects:

- **“Functional Overview” on page 201**
- **“Macro Creation Design Flow” on page 202**
- **“Using a Macro in a Design” on page 207**
- **“Warning and Error Messages” on page 222**

17.1 Functional Overview

The user-defined macro creation capability in QuickWorks allows designers to create macros of any size with placement and routing information embedded in it. Designers should be able to reuse their IP, generate a library of some standard modules, which can be picked up and used in any design or even used as ready-made IP developed by any third party.

This feature allows a designer/IP developer to develop a design on QuickLogic devices and store the final implementation as a macro. This macro can be used in other designs, just by adding the macro symbol in the design schematic or by instantiating it in design HDL. This provides an already synthesized, mapped placed and routed netlist for a macro instance.

Along with a macro, the post-layout simulation module can be provided that can be used during the design post-layout simulation.

17.1.1 Terms and Definitions

The following are descriptions of some terms referred to throughout this section:

- **HDL**—Hardware Description Language (includes Verilog and VHDL)
- **SYM**—symbol file
- **QCF**—QuickLogic Constraint File
- **Secured Macros**—IP-protected macros. Physical implementation of such macros is hidden from the end user.

For more information, see also:

- *QuickWorks Tutorial User Manual – Section: Macro Tutorial*

© 2008 QuickLogic Corporation

17.2 Macro Creation Design Flow

This section provides guidelines for the creation of macros using SpDE.

17.2.1 Setting up SpDE for Creating Macros

To create a macro, SpDE should be setup in *Macro Design Mode*, which is different from the normal operation mode of SpDE. By default, when SpDE is installed, it starts in normal mode. A typical user (except a macro designer) uses this mode to implement the designs and to save a `*.chp` file used for device programming.

Macro Design Mode in SpDE can be enabled by:

- Selecting **Tools>Macro Creation Mode.**
- Clicking on the toolbar icon for Macro Creation Mode.
- Using the **Ctrl+Alt+M** shortcut key.

17.2.2 Importing Design into SpDE

Just like in normal flow, in Macro Design Mode, design files in QDIF/EDIF or CHIP file format can be imported into SpDE and saved as macros.

NOTE: Verilog/VHDL designs can be exported to QDIF/EDIF files using the Synthesis tool and then imported into SpDE.

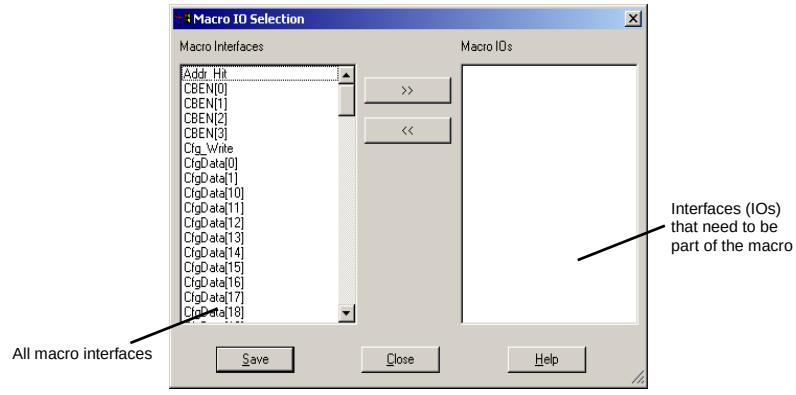
NOTE: Design in Schematics can either be exported as QDIF or as Verilog/VHDL netlist. The resulting Verilog/VHDL can be synthesized to generate a QDIF/EDIF netlist, which can then be imported into SpDE.

The following are some of the limitations of the import process:

- Only LOGIC, RAM, ECU and input/output pad cells can be used in Macro Design.
- While creating a macro on an ESP device, designs interacting with ESP blocks cannot be saved as a macro.
- PLL and DDRs are not supported in macro design.

17.2.3 Macro IO Selection

This selection dialog is used to select the IO that needs to be part of the macro.



© 2008 QuickLogic Corporation

- If you select IOs in the macro then this macro design can only be used in the same device and package in which it is created.
- Macro Interfaces lists all the inferfaces of the macro.

17.2.4 Processing Macro Design

In Macro Design Mode, the Sequencer tool is disabled. A simulator can be chosen be selecting **Tools Options>Back Annotation**.

17.2.4.1 Automatic Selection of Window

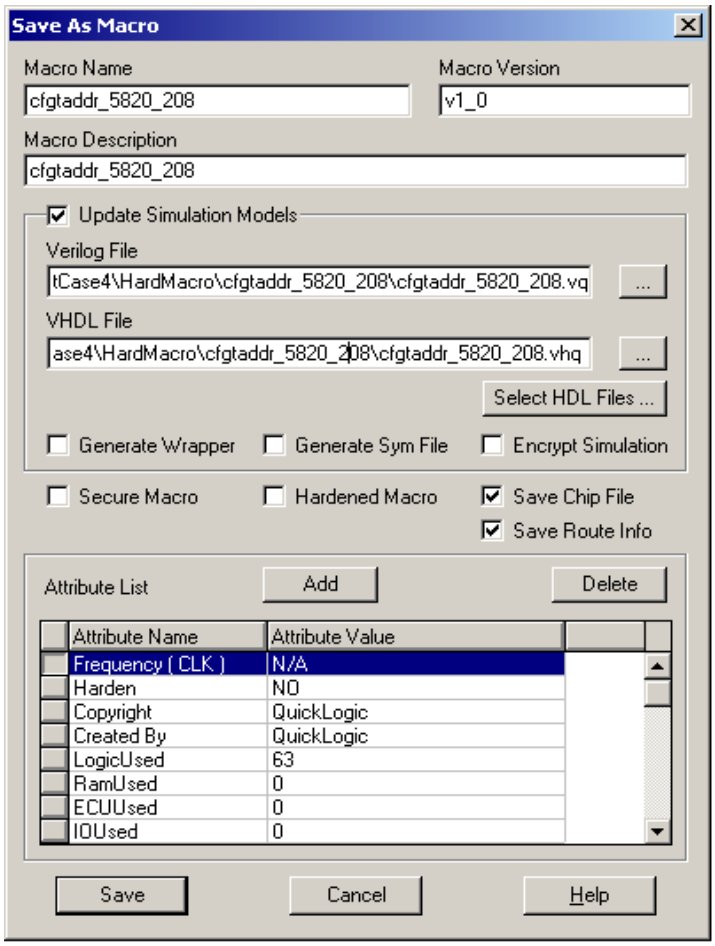
To give macros a regular shape, SpDE automatically selects a window to fit the macro design in one corner of the device. To turn off this feature in SpDE, add the following in the `SpDE.ini` file.

```
[placer]

;Auto Selection of Window in Macro Design Mode [TRUE (default) ,
FALSE : to turn off]

AutoSelectWindow=FALSE
```

NOTE: It is advisable to define a window through Constraint Manager to make sure that macro has a regular shape. For the creation of pre-routed macros which use only logic cells, it is advisable to select the window two rows and columns away from IO cells.

17.2.5 Save Design as Macro

After running tools and making sure that a design meets all of the requirements, it can be saved as a macro by selecting **Tools>Save As Macro**. If there are no verifier or error messages for the macro design, the following dialog box opens.



NOTE: Verifier error messages are listed in the transcript window. For warnings and errors in macro design mode, refer to **Appendix D, "Error Messages" on page 529**.

17.2.5.1 Save As Macro File Attributes

The following are attributes of the `Save As Macro` command:

Table 17-1: Save As Macro File Attributes

Attribute	Description
Macro Name	Macro Name is read from the `design.qcf` file, if it exists, otherwise the design name is used. (e.g., count16)
Macro Version	The version number of the macro. (e.g., v1_0)
Macro Description	Information related to bits, speed, ram bits etc., can be entered in this text box. (e.g., Counter 16 Macro)

© 2008 QuickLogic Corporation

Table 17-1: Save As Macro File Attributes

Attribute	Description
Update Simulation Models Information	If this checkbox is enabled, the core simulation model information can be updated.
Verilog File	The Functional Verilog Simulation model file with path. This file is copied to the version directory where the macro is saved in munged form.
VHDL File	The Functional VHDL Simulation model file with path. This file is copied to the version directory where the Macro is saved in munged form.
Select HDL files	This button invokes the HDL File Selection dialog box. Verilog or VHDL files to be included in the core simulation model can be selected.
Generate Wrapper	The wrapper file is an HDL file that defines the macro interface and contains the simulation model for the macro. Check this option to generate a wrapper file. This wrapper file can further be used to either simulate the macro or use it as part of any other design.
Generate Sym file	Check this option to generate a symbol file for the macro. This symbol file can be used to create an instance of this macro in any other schematic design.
Encrypt Simulation File	This option lets you encrypt the core simulation model to provide an additional security.
Secure Macro	If the macro is declared as secure, then all data files for the macro are encrypted and contents of the macro are hidden from the macro user. This option provides security to macro Intellectual Property.
Hardened Macro	If the macro is declared as hardened, its location is fixed and cannot be moved in the target device and package. The hardened macro can be used only in the device and package where it has been created.
Save Chip File	Once the macro file is saved, the design closes automatically. To restart from the current stage of design, it is required to save the chip file. Select this check box to ensure chip file is saved.
Save Route Info	Check this option to save the routing information of the macro. By default this option is checked.
Attributes List	SpDE allows users to define attributes for the macro design. There are certain default attributes that are displayed in the dialog box such as *Copyright, Created By, LogicUsed*, *RamUsed, QmathUsed* and *Frequency.* Attributes *Copyright* and *Created By* can be edited. None of the mandatory attributes can be deleted. Extra attributes can de added, deleted or modified using the following options: • **Add**—SpDE opens Add Attribute dialog box, if the **Add** button is clicked in the Save As Macro dialog box. New attributes and values can be entered using this dialog box. • **Delete**—an attribute can be deleted using the **Delete** button. Select the row to be deleted from the attribute list table and click the Delete button. • **Save**—this command saves and installs the macro in a particular directory version with all settings automatically.

The following directory setup is created in /pasic/spde/data directory to store the macro:

- /pasic/spde/data/macro/<macroname>/<versionname>/macrofiles (i.e., `design.mcr` files, core simulation models.)

Wrapper and .vh files are generated and copied into the macro name directory. Symbol files are generated and copied into /pasic/lib/macro directory.

© 2008 QuickLogic Corporation

Alternatively, if you want to save the macro data files to a different path the following settings in the spde.ini file should be done.

[HardMacro]

MacroPathName=<path>

17.2.6 Setting Constraints for Creating a Macro

The following are the constraints that can be used through the `design.qcf` file:

- Macro Name—name of the macro

  Usage: `macro name <nameofthemacro>`

  NOTE: For non-alphanumeric version, enclose the name string in double quotes.

- Version—attaches a version string to the saved for macro.

  Usage: `macro version v1_0`

  NOTE: For non-alphanumeric version, enclose the version string in double quotes.

- Macro attributes—alternatively you can specify the macro attributes through qcf file using the following syntax

  Usage: `macroattribute "<attributeName>" "attributevalue"`

- Macro IO Interface Constraint–This constraint is used to select macro io for macro design. Normal Macro Interfaces are defined without – pin option.

  Usage: `set macro int net -pin "interfacenetname"`

  Usage: `set macro int net "interfacenetname"`

For more information, see also:

- *QuickWorks Tutorial User Manual – Section: Macro Tutorial*

© 2008 QuickLogic Corporation

17.3 Using a Macro in a Design

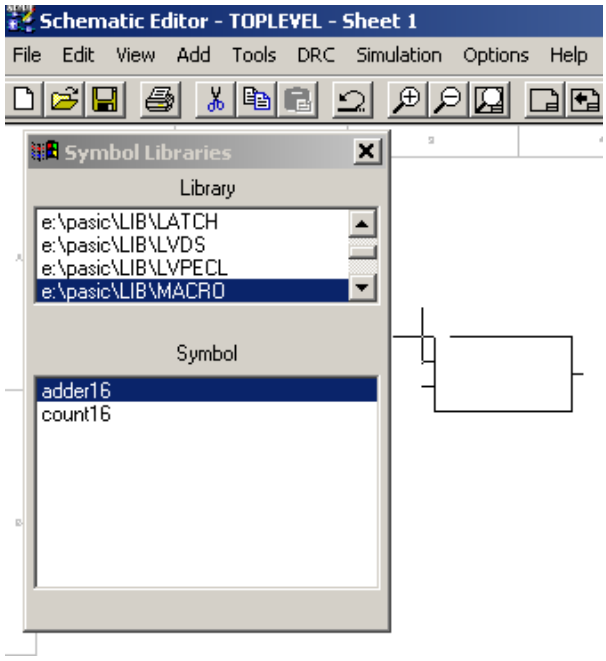
This section describes how macros are used in designs.

By default, SpDE is opened in normal mode. There is no special setting for using macros. It can be used as any other design block.

17.3.1 Creating Designs Using Macros

17.3.1.1 Schematic Based Flow

The SCS Schematic Editor is used to capture and create schematics in a design. Although multi-sheet schematics can be created, this is generally discouraged as they indicate a flat design style rather than a hierarchical one. The editor enables easy navigation through a hierarchical design.



You can perform the following operations in the Schematic Editor:

- Add symbols and wires—invokes a symbol libraries dialog box which can be used to select the macro libraries.
- Add instance names to all symbols (optional)—iterated instances allow a single instance of a symbol to represent many identical instances in a parallel connection. This feature is useful when connecting an array of identical macros to a bus.
- Add net names—power and ground connections are made in the Schematic Editor by naming the net a VCC or GND, respectively.
- Add I/O markers to input and output nets.
- Run the schematic checker.
- Create a symbol for the schematic, if it is not a top-level design.
- Save the design.

© 2008 QuickLogic Corporation

- Export the design to QDIF/Verilog/VHDL—synthesize the design, if exported to Verilog or VHDL, and load the QDIF/EDIF file in the SpDE tool.

17.3.1.2 HDL Based Flow

To enter a HDL design:

1. Write Verilog/VHDL source code in Turbo Writer.
2. Verify the syntax using Precision Syntax Check.
3. In case of Verilog, include the wrapper file for each macro using `'include` statement located in the pasic/spde/data/macro or user define macro data directory, if not done automatically.
4. Create Input Vectors (Verilog Test-Fixture)—from within Turbo Writer, a Verilog test-fixture is created either manually or with the graphical Waveform Editor.

To perform a prelayout simulation:

1. Functionally simulate the Verilog/VHDL design.
2. The simulation results can be viewed using the Data Analyzer.
3. The Verilog/VHDL file is exported to SpDE.

To perform Synthesize and Place-and-Route:

1. Synthesize the Verilog/VHDL design in Precision RTL.
2. When the Precision RTL compilation is complete, a netlist is created.
3. The design is optimized, placed & routed. The Delay Modeler generates precise postlayout delays, which are back-annotated for simulation.

17.3.2 Functional Simulation

While simulating the design using Macros, the following files (i.e., wrapper and Core Simulation Model) should be included in the simulation script:

In case of Verilog:

```
/pasic/spde/data/macro/<macroname>/macroname.v
/pasic/spde/data/macro/<macroname>/<versionname>/macroname_design.v
or
<userdefined>/macro/<macroname>/macroname.v
<userdefined>/macro/<macroname>/<versionname>/macroname design.v
```

In case VHDL:

```
/pasic/spde/data/macro/<macroname>/macroname.vhd
/pasic/spde/data/macro/<macroname>/<versionname>/macroname_design.vhd
or
<userdefined>/macro/<macroname>/macroname.vhd
<userdefined>/macro/<macroname>/<versionname>/macroname design.vhd
```


© 2008 QuickLogic Corporation

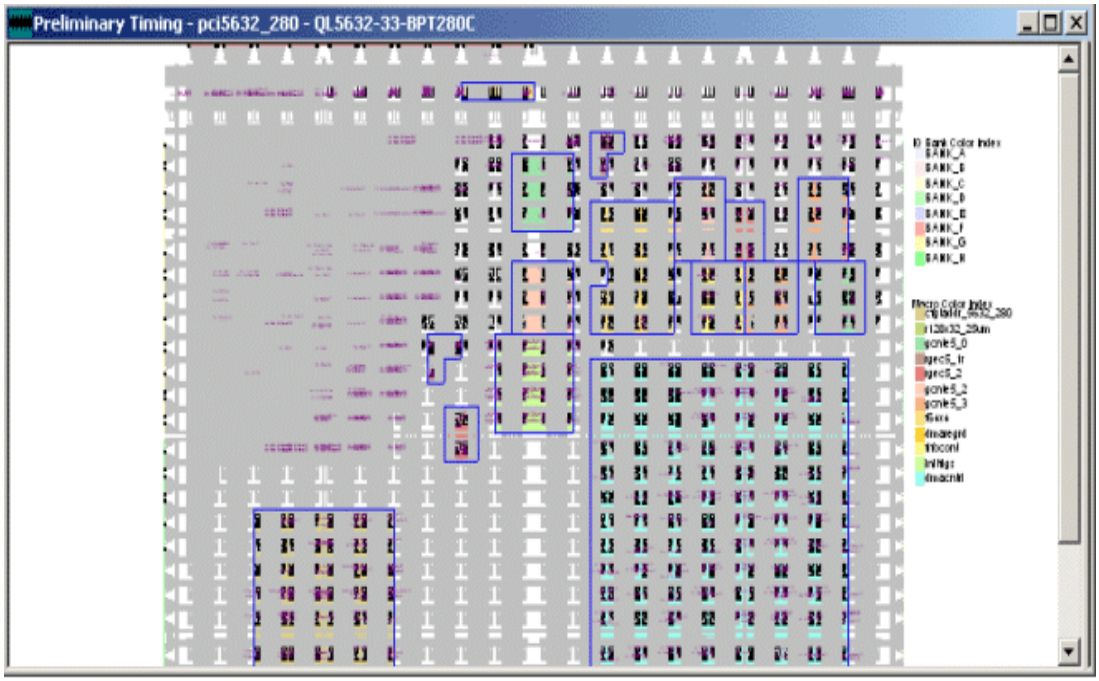
17.3.3 Setting Up SpDE for Using Macros

By default, when SpDE is installed, it starts in normal mode. A typical user (except a macro designer) uses this mode to implement the designs and to save a `*.chp` file used for device programming. Ensure that SpDE is not in Macro Design Mode, if the current design is not intended to be saved as macro.

17.3.4 Importing Design (Macro + User Design) in SpDE

Verilog/VHDL designs can be exported to QDIF/EDIF using the Synthesis tool and then imported in SpDE. Schematic designs exported to Verilog/VHDL should follow the same flow, and designs exported to QDIF directly can be imported in SpDE.

In case of pre-placed macros, while importing a design using macros, SpDE assigns placement for macro instances that are fixed or hardened. If the placer tool is run, all the other instances are also placed. The following screen capture shows the physical layout for a design using pre-placed macros that are fixed, just after it has been loaded into SpDE.



The following convention is followed to show macro instances in the physical view:

- Each macro has its own assigned color. Color index for a macro is shown on the right side of the physical view. Instances of the same macro are assigned the same color.
- Logic part of a macro instance is shown with a boundary across the grouped logic cells of the instance. ECU and RAM cells are also shown in the same way but without grouping.
- Fixed instances are shown with blue boundaries. Unfixed instances are shown with black boundaries.
- Instances are shown with the background of the assigned color of the macro to which they belong. In secured instances this coloring hides cell and route resources.
- Interface ports are shown on the instances as circles and port names.

© 2008 QuickLogic Corporation

- An instance is selected or highlighted by clicking on any point inside the instance. The selected instance is shown with dotted boundary. When an instance is selected, the following attributes of the instance are displayed in the macro information tab of the transcript window:
  - Name and version of the macro to which the selected instance belong.
  - Whether the macro to which the instance belongs is secured/routed/hardened or not.
  - Name of the selected instance.
  - Current orientation of the instance.
  - Whether position of instance is fixed or not.
  - Whether orientation of instance is fixed or not.

  NOTE: If the selected instance does not contain the logic part, the last three attributes are not displayed.

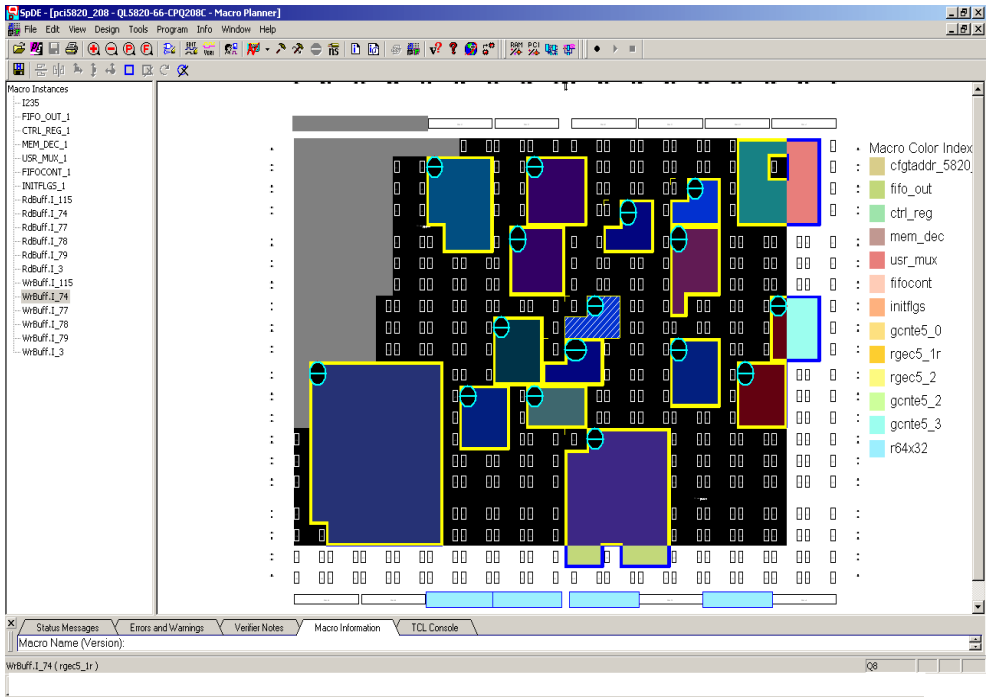
If SpDE is not able to find a location for any of the macros, it informs the user via error message (see **Appendix D, "Error Messages" on page 529**). In that case the user design needs to be modified to remove some of the macros or retarget the design to a bigger Quicklogic part. It is possible to change the placement and orientation of macros using the Macro Planner tool (see **"Macro Planner" on page 212**.)

The SpDE placement tool automatically finds an optimal placement and orientation for (unfixed) macros, in case the user does not place them manually. If the placement tool cannot find an optimal placement, the user is advised to place all macros manually using Macro Planner to achieve a good placement.

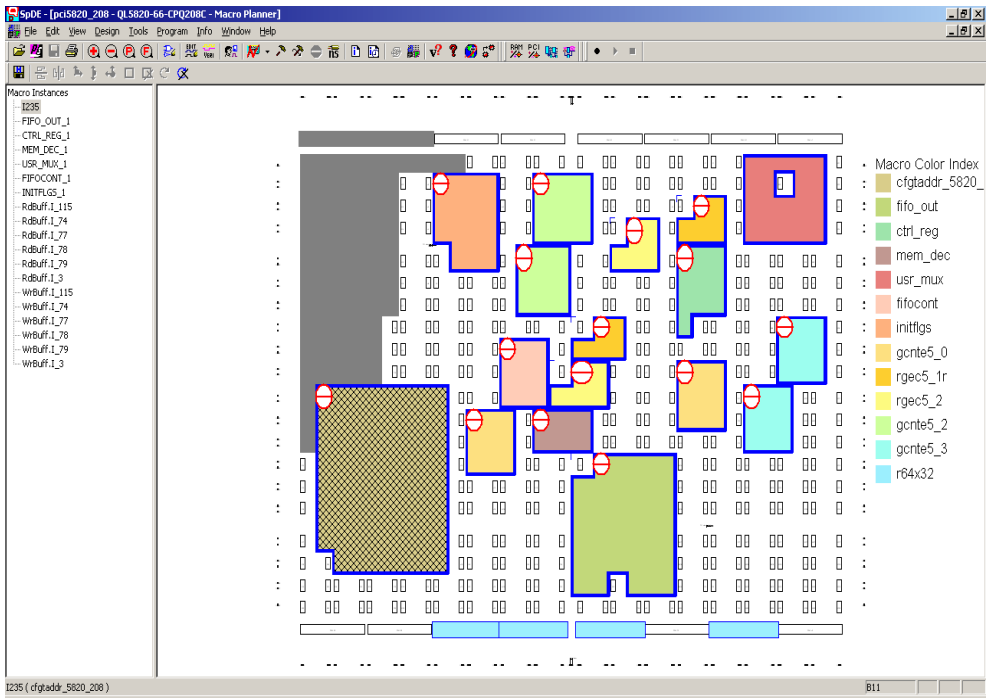
In case of pre-routed macros, while importing the design using macros in SpDE, it finds the valid locations for macros during the design load stage. The macro instances will not fix to any particular location. The HM planner allows you to place the pre-routed macros.

© 2008 QuickLogic Corporation

Placing macros: When you click on a macro instance, the HM planner will show all valid locations for this pre-routed macro that can fit in the black color. Cells colored black indicates that the macro can fit in that location. See the figure below:



The figure below shows a hardened macro instance selection. There will not be any valid location for this case.



NOTE: In certain cases the time taken to find possible valid locations for pre-routed macros will be longer. To turn this option off you can select the following spde.ini option in the router section, see below.

[router]

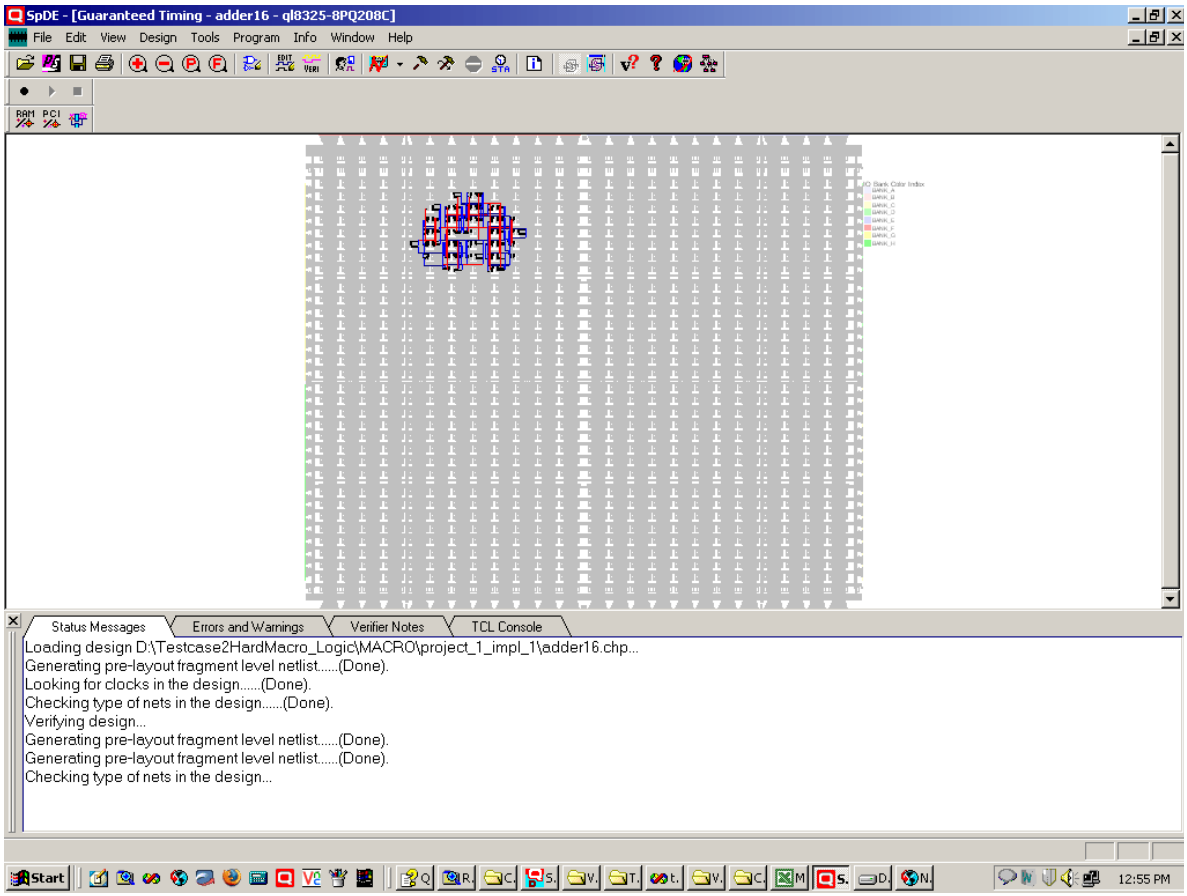
SkipFindLoc=1

In this case you can place the macro in all the locations. The router tool will validate this location and issue warning messages if it is not able to route the design.

NOTE: Operations such as flip, rotation are not allowed for pre-routed macros.

17.3.5 Running Tools

After all macros are placed in a valid location, select **Tools->Run** to run the tools and save the chip file.


SpDE - [Guaranteed Timing - adder16 - ql8325-8PQ208C]
File Edit View Design Tools Program Info Window Help
Status Messages
Errors and Warnings
Verifier Notes
TCL Console
Loading design D:\Testcase2HardMacro_Logic\MACRO\project_1_impl_1\adder16.chp...
Generating pre-layout fragment level netlist.....(Done).
Looking for clocks in the design.....(Done).
Checking type of nets in the design.....(Done).
Verifying design...
Generating pre-layout fragment level netlist.....(Done).
Generating pre-layout fragment level netlist.....(Done).
Checking type of nets in the design...
Start
12:55 PM

17.3.6 Macro Planner

The Macro Planner tool enables users to change, fix and unfix macro instance position and orientation of every macro. By using this tool, designer can achieve desired results in terms of FPGA area and timing performance.

© 2008 QuickLogic Corporation

Macro Planner offers the flexibility to place various macro instances of a design close together or far apart with different orientations, so that the intended macro design specifications are met. The function provided by the macro planner is commonly referred to as *floor planning* in ASIC design.

Providing a control over placement of various macro instances ensures that the designer's knowledge of the design structure is utilized to meet, or even exceed, the chip space and performance thresholds. This tool is especially suitable for those designs, where it is crucial to improve critical path delays and to make optimal use of the silicon area. The macro placement planning can either be done using Macro Planner or through a constraint specified in the `qcf` file.

NOTE: The `qcf` file is the QuickLogic's constraints file, containing information such as pin constraints, flip-flop placement, and timing constraints.

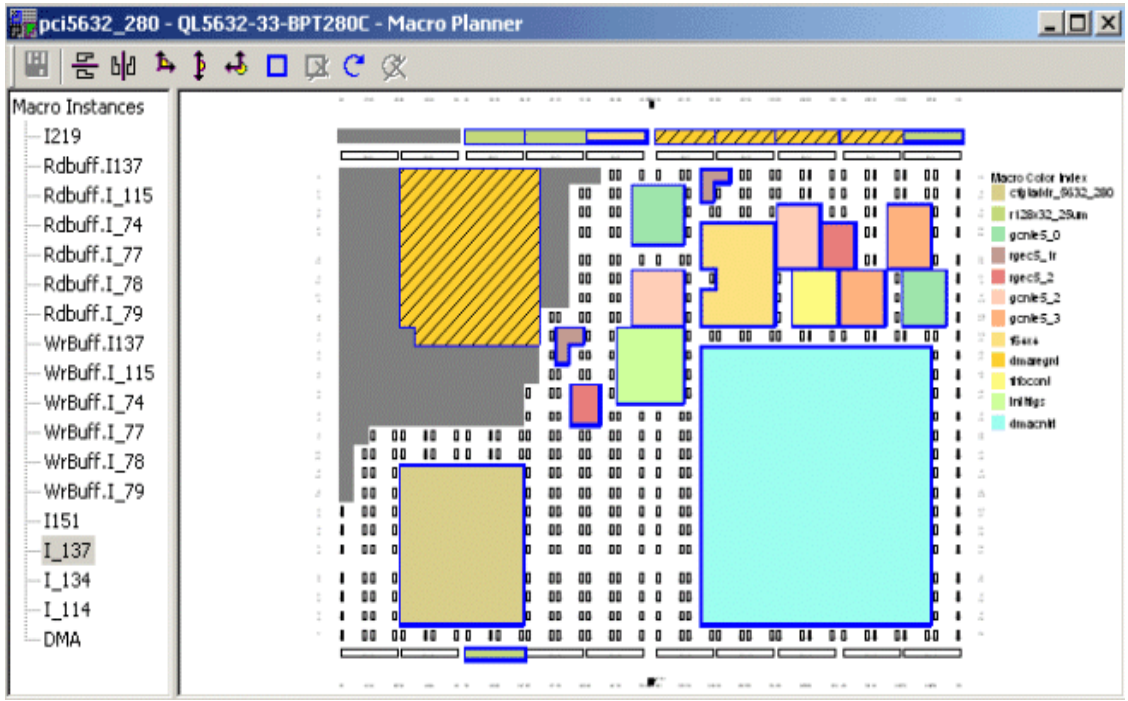
17.3.7 Using Macro Planner

17.3.7.1 Macro Planner View

This view is a simplified view of the physical viewer. The logic, RAM, ECU, clock, IO and mux cells are displayed as squares and rectangles. Instances of different macros are shown in different colors. Color index scheme on the right of the view shows the colors assigned to each macro. An instance with fixed position is shown with a thick border around the LOGIC cell portion (referred to as the *logic part*). An instance with fixed orientation is shown with a circle on the top left. A highlighted instance is shown with a hatch pattern on it. If the highlighted instance belongs to a hardened macro then it is shown with a cross hatch pattern on it.

© 2008 QuickLogic Corporation

For easy browsing of the instances in the design. an explorer panel is provided containing a tree control similar to that in the MS Windows Explorer, showing the list of macro instances.



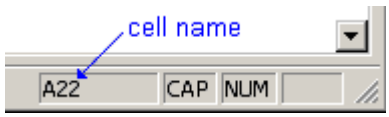
17.3.7.2 Invoking the Macro Planner View

After loading the design with a hard macro. The planner view should be invoked by:

- Selecting **Tools>Macro Planner.**
- Clicking on the toolbar button .
- Using the shortcut key **Ctrl+Alt+P.**

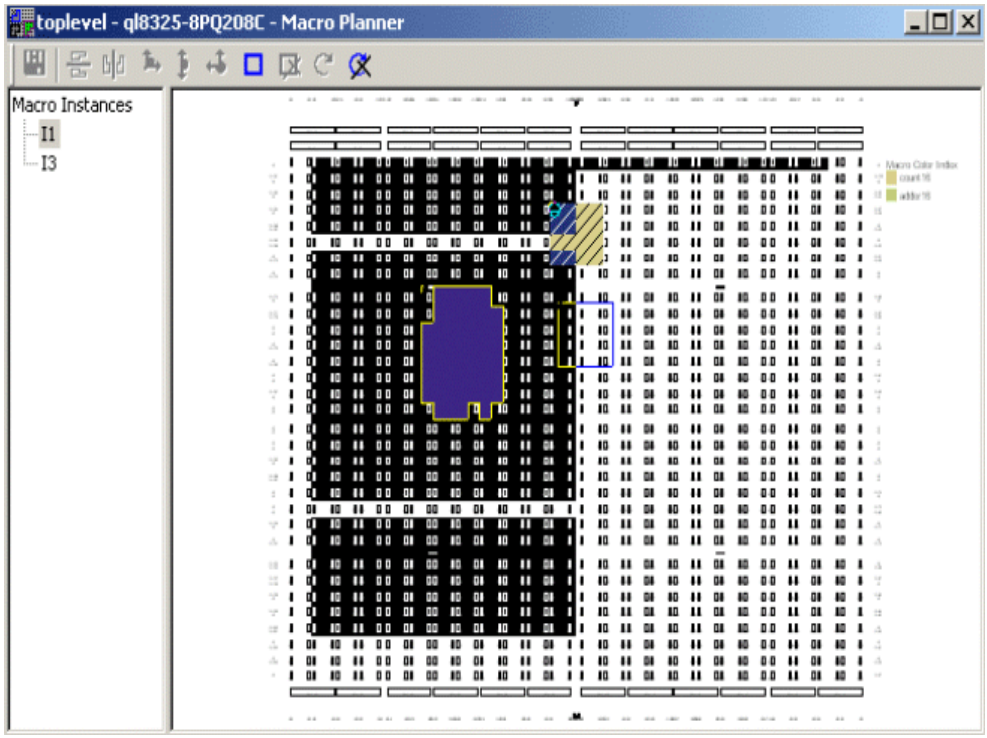
17.3.7.3 Moving Instances

The logic part of an instance can be moved across the logic cell array using the mouse. Only instances that are not fixed or hardened can be moved. The ECU and RAM cell part of an instance can be moved independently of the logic part. Unlike a logic part, which can be moved only as a group, RAM and ECU parts can be moved one cell at a time. While moving an instance, its top left cell name can be viewed in the status bar and the same cell is shown as blinking.



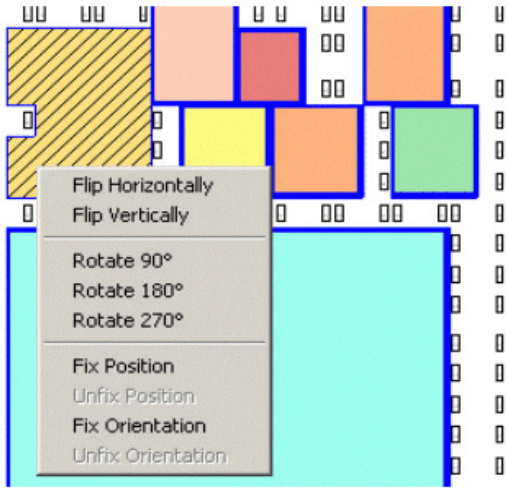
© 2008 QuickLogic Corporation

If a logic part of an instance is being moved, the status bar shows the name of the top left cell of the logic part of the macro instance. For an instance which is routed, all the cell locations which are valid i.e., to be the top left cell for the logic part of the instance which is being moved, are shown in a black (highlighted) color. If an ECU or RAM cell of the instance is being moved, the status bar shows the name of the cell.

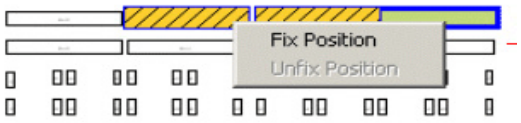


17.3.7.4 Macro Planner Menu

The drop-down menu is invoked by right-clicking on the logic, RAM, or ECU parts. For the logic part of the instance, there are options for flipping, rotating and fixing/unfixing of the position and orientation.

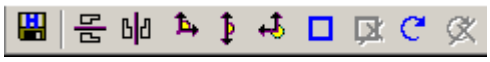


For the RAM and ECU parts, there are only two operations possible: fixing and unfixing of the position. If the RAM and ECU parts belong to a hardened macro then fixing and unfixing operations are not possible for those parts.



17.3.7.5 Macro Planner Toolbar

As an alternative, the operation on the macro instances can be performed using the toolbar that appears on the top of the planner view.

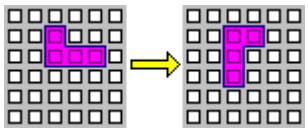


NOTE: Enable or disable status of the icon in the toolbar changes according to the status of the macro instance.

© 2008 QuickLogic Corporation

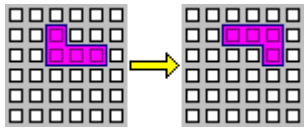
17.3.7.6 Rotate 90 Degrees

This command changes the orientation of an instance by rotating it 90 degrees clockwise. The width and height are interchanged but the x- and y-origins remain the same.



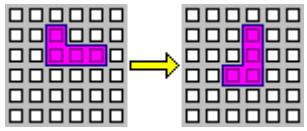
17.3.7.7 Rotate 180 Degrees

This command changes the orientation of an instance by rotating it 180 degrees clockwise. The width and height are interchanged but the x- and y-origins remain the same.



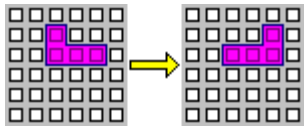
17.3.7.8 Rotate 270 Degrees

This command changes the orientation of an instance by rotating it 270 degrees clockwise. This rotation is equivalent to rotating the instance 90 degrees anti-clockwise. The width and height are interchanged but the x- and y-origins remain the same.



17.3.7.9 Flip Horizontally

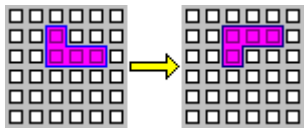
This command changes the orientation of an instance by flipping it horizontally. The width and height are interchanged but the x- and y-origins remain the same.



© 2008 QuickLogic Corporation

17.3.7.10 Flip Vertically

This command changes the orientation of an instance by flipping it vertically. The width and height are interchanged but the x- and y-origins remain the same.



17.3.7.11 Fix Position

The logic part of an instance can be fixed at its current location. If the instance to be fixed overlaps with other fixed instances, it can be fixed after unfixing the overlapping fixed instances. If the instance is over the sapphire area, it is not possible to fix it. Fixed instances cannot be moved. Unlike logic part, RAM and ECU parts are fixed one cell at a time. If the logic optimizer is not run, fix or unfix is not possible for the RAM and ECU.

17.3.7.12 Unfix Position

An instance can be unfixed from the current location so that it can be moved to a different location. Unlike logic part, RAM and ECU parts are unfixed one cell at a time.

17.3.7.13 Fix Orientation

Fixing the orientation restricts the user to flipping or rotating the logic part of an instance. This operation is not available for the ECU and RAM cells of the instance.

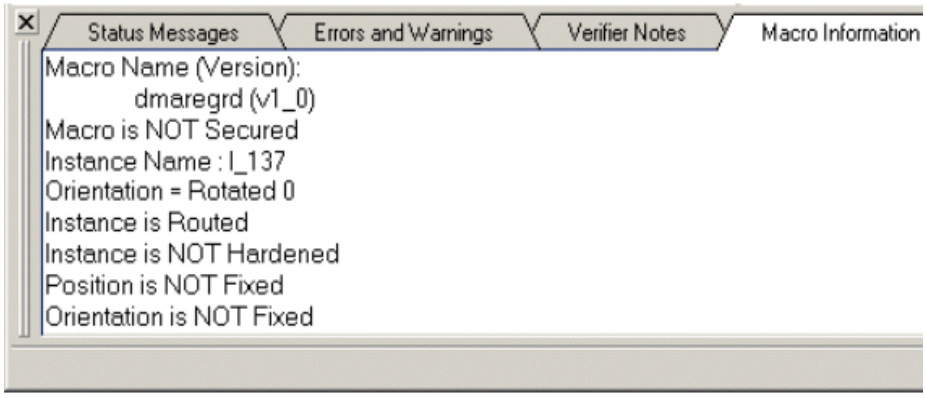
17.3.7.14 Unfix Orientation

The orientation of an instance can be unfixed so that flipping or rotating can be performed. This operation is not available for the ECU and RAM cells of the instance.

© 2008 QuickLogic Corporation

17.3.7.15 Highlighting Macro Instance

Instance becomes highlighted when clicked on with the mouse. The highlighted instance is differentiated from other instances by having a cross hatch pattern drawn on it. Alternatively, an instance can be selected by clicking on the instance browser on the left of the macro planner view. Attributes or current state of the highlighted instance are displayed in the Macro Information tab of the transcript window.



The following information is displayed in the Macro Information tab:

- Name and version of the macro to which the selected instance belong.
- Whether the macro to which the instance belong is secured/routed/hardened or not.
- Name of the selected instance.
- Current orientation of the instance.
- Whether position of instance is fixed or not.
- Whether orientation of instance is fixed or not.

If the selected instance does not contain the logic part, the last three items are not displayed.

17.3.7.16 Saving Placement

The changed placement and orientation of the instance can be saved through the menu **Tools>Save Macro Planner**, the toolbar button, or the shortcut key **Ctrl+Alt+S**. This menu or button becomes enabled only when any instance is fixed or unfixed. Macro planner is closed after the placement is saved. The physical viewer is changed according to the new macro instance placement.

For more information, see also:

- *QuickWorks Tutorial User Manual – Section: Macro Tutorial*

17.3.8 Macro Fix Placement Constraints

The macro packer/placer automatically determines the placement as well as orientation for all macro instances. However, sometimes the designer may wish to place some macro instances at a certain place in the chip layout with different orientation to get optimal results. To support this, SpDE enables the user to put a constraint on macro instances using the `.qcf` file. SpDE flow does not modify these user-assigned constraints.

© 2008 QuickLogic Corporation

The easiest way to perform fix placement for macro instances is by using the Macro Planner tool (for more details see **"Macro Planner" on page 212**).

17.3.8.1 Macro Orientation

To get compact placement and achieve timing requirements of a design, the following two types of orientations are supported in SpDE:

- Rotation of macro instance
- Flipping of macro instance

NOTE: These two orientations are supported only for logic cells which are part of the macro but not for RAM/ ECU cells.

- Rotation of macro instance—four types of rotation are supported on each macro instance:
  - 0 degree rotation (0)
  - 90 degree rotation (90)
  - 180 degree rotation (180)
  - 270 degree rotation (270)
- Flipping of macro instance—two types of flipping are supported on each macro instance:
  - Horizontal flipping (H)
  - Vertical flipping (V)

NOTE: Combined constraint for flipping and rotation is also supported on each macro Instance.

17.3.8.2 Macro Constraint Syntax

A macro instance with different orientation can be placed in a pre-selected region of the chip, using the upper-left corner logic cell name, called the *reference cell*, and the macro orientation.

Syntax to place a macro instance in the `<design>.qcf` file:

```
macroplacement <Macro Instance Name> <Reference Cell Name>  [flip <flip type>]
[rotate <rotation type>].
```

where:

- Macro Instance Name—specifies the macro instance to be placed.
- Reference Cell Name—specifies the logic cell name, where the macro instance should be placed within a chip. After validating the reference cell, left-top point of the rectangle containing macro is moved to the specified cell.
- Flip type—can be one of the following two:
  - H—horizontal flipping
  - V—vertical flipping
- Rotation type—It can be one of the following four:
  - 0—0 degree rotation
  - 90—90 degree rotation
  - 180—180 degree rotation
  - 270—270 degree rotation

© 2008 QuickLogic Corporation

NOTE: In the above syntax, specifying Macro Instance name and Reference Cell Name is required, while specifying flipping and rotation is optional.

NOTE: **flip** and **rotate** are keywords.

17.3.8.2.1 Constraints for RAM/ECUs Gates

If macro is using RAM/ECU gates, these can be individually placed using **placeram**/**placeecu** command (see **"Fix Placement Using the QCF File" on page 306**). If the logic part of the macro instance is fixed and individual RAM/ECUs gates of the instance are not fixed, then all unfixed RAM/ECU gates are considered as fixed.

17.3.8.2.2 RAM/ECUs In Macro

If both the macro instance and the RAM/ECU gates are fixed, then higher priority is given to individual placements. If the macro instance is fixed and individual RAM/ECUs gates are not fixed, then all unfixed RAM/ECU gates follow the macro instance fix placement constraint.

Macro Fix Placement Example

For example, a user design `top_level.qdf` has the following macros and their instances:

- Macro Name: `adder16a.qdf`.
  - Description: 16-bit adder
  - Instances: This macro has the following instance:
    - I7
- Macro Name: `count16.qdf`:
  - Description: 16-bit counter
  - Instances: This macro has the following two instances:
    - I5
    - I6

Example:

- Adder16a (16-bit adder) macro instance I7 needs to be placed at Logic Cell P9 with default orientation. The command format in the `.qcf` is:

```
macroplacement I7 P9
```

- Count16 (16 bit counter) macro instance I6 needs to be placed at Logic Cell J13 with vertical flipping and 90 degree rotation. The command format in the `.qcf` is:

```
macroplacement I6 J13 flip V rotate 90
```

17.3.9 Processing Design

In SpDE's normal mode, after the design is loaded, it enables all tools to be run and the chip file to be saved. The Delay Modeler generates precise post layout delays, which are back annotated for simulation. If the design uses secured macros then the chip file will be encrypted.

For more information, see also:

- *QuickWorks Tutorial User Manual – Section: Macro Tutorial*

© 2008 QuickLogic Corporation

17.4 Warning and Error Messages

17.4.1 Fatal Errors

Table 17-2 describes fatal error messages generated by the macros.

Table 17-2: Fatal Error Messages

FATAL ERROR: Macro:<> INPUT <> of Instance <>, is floating
A macro instance is found in the design with its input floating. All the macro inputs need to be tied either to default (VCC/GND) or to a valid signal.
FATAL ERROR: PLL cells are not supported in Macro
Only LOGIC RAM, ECU and plain I/Os can be part of a design in macro creation mode. PLL cells are not supported.
FATAL ERROR: BIDIR Cell usage of gate <>, is not supported in Macro
BIDIR cells can only be used as plain input/output pads, any other type of usage (e.g., flip flop, bi-directional signal, etc.) is not supported in the macro creation mode.
FATAL ERROR: <> Cell usage of gate <>, is not supported in Macro
Only plain input/output pads are supported in the macro mode. The <> Cell Usage of gate <> is not plain.
FATAL ERROR: Macro without Logic, RAM and ECU usage
This macro is empty (macros should have some usage of LOGIC/RAM/ECU cells).
FATAL ERROR: Design connected to ESP cannot be saved as Macro
A design cannot have any connections to ESP while creating a macro on an ESP device.

17.4.2 Errors

Table 17-3 describes error messages generated by the macros.

Table 17-3: Error Messages

ERROR: No Space for <> Macro Instance
The targeted device is insufficient to pack all macros. Target for bigger device or try packing using macro constraint in the `.qcf` file with a different orientation.
ERROR: Macro Library does not exist
This implies that macro is not yet build. Build the macro using macro creation mode.
ERROR: Term Name not found for Macro Interface Net <>
There is a different macro with the same name and version. Build the macro which is used in the top level design.
ERROR: Could not load macro constraint file
The macro constraint file cannot be loaded.
ERROR: Macro Core (<>) is not supported in this device (<>)
The Macro Core device and the targeted device are incompatible.
ERROR: Macro Core uses RAM Blocks, Hence it is not supported in this device (<>)
The Macro Core device and the targeted device are incompatible.

© 2008 QuickLogic Corporation

Table 17-3: Error Messages (Continued)

ERROR: Multiple Term Interface for Macro Net <>
There are some nets which are connecting more than once to macro I/Os. Try to separate them using buffers.
ERROR: Cells are not allowed in Macro Design
These types of cells are not supported in macro design.
ERROR: Macro with <> cell not supported
These types of cells are not supported in macro mode.
ERROR: Macro: Unable To Pack Design in Targeted Device
The targeted device is insufficient to pack all macros.
ERROR: Empty Macro Instance <>
There is nothing in the macro instance.
ERROR: Device Insufficient to pack given Macro <>
Target for bigger device or try packing using macro constraint in the `.qcf` file with a different orientation.
ERROR: Invalid orientation: unknown orientation for macro
This orientation is not supported. For more information contact Quicklogic (**www.quicklogic.com**).

17.4.3 Warnings

Table 17-4 describes warning messages generated by the macros

Table 17-4: Warning Messages

WARNING: Macro:<> OUTPUT <> of Instance <>
The output term of the instance is not connected to any logic.
WARNING: Registered BIDIR Cell Output (<>) is not supported in Macro
Only Plain BIDIR Cell usage is supported in macro mode.
WARNING: Macro Instance <> cant be placed at <> Reference Cell. This Area is already used by other Macro/design
The target macro area is already used by another macro or a user design using fix placement constraint. To place this macro in the same location, undo the other macro/user design that is fixed into this macro area.
WARNING: Macro Directory not found in /pasic/lib,<>.sym not generated
This warning message occurs when macro directory is not available in the /pasic/lib directory.
WARNING: Design cannot be saved as Macro without placement tool run
This warning message implies that a macro file cannot be saved if the placer tool is not run.
WARNING: pull_ff_into_io <> constraint is not supported in Macro Design Mode - Constraints Ignored
This constraint is not supported in macro design mode.
WARNING: Macro <> doesn't use any logic cell, Instance <> can not be fixed
This warning message implies that macro does not use any logic cells. The macro fix placement is supported to those macros which uses at least one logic cell.
WARNING: Macro Instance <> can not be placed at cell <> with Orientation <>

© 2008 QuickLogic Corporation

Table 17-4: Warning Messages (Continued)

The target macro area with orientation is already used by another macro or a user design using fix placement constraint. To place this macro in the same location, undo other macro/user design that is fixed into this macro area.
WARNING: Unable to fit Macros. Please floorplan them manually
SpDE placement tool cannot find an optimal placement due to device size limitations, place macros using the Macro Planner tool to achieve a good placement.

© 2008 QuickLogic Corporation

Chapter 18
Creating RAM, FIFO, and ROM Modules

This chapter discusses the following aspects of creating RAM, FIFO, and ROM Modules:

- **“Functional Overview” on page 225**
- **“Creating RAM, FIFO and ROM Modules for Pinnacle, Pre-Pinnacle and Eclipse Devices” on page 225**
- **“Creating RAM and FIFO Modules for PolarPro and PolarPro II Devices” on page 244**

18.1 Functional Overview

The QuickLogic RAM/ ROM/ FIFO Wizard tool can automatically configure QuickLogic embedded RAM blocks as RAMs, ROMs, or FIFOs. Both Verilog and VHDL blocks can be created for instantiation in an HDL design. The port lists necessary for creating schematic symbols are written to the HDL files, allowing you to create schematic symbols of the same macro.

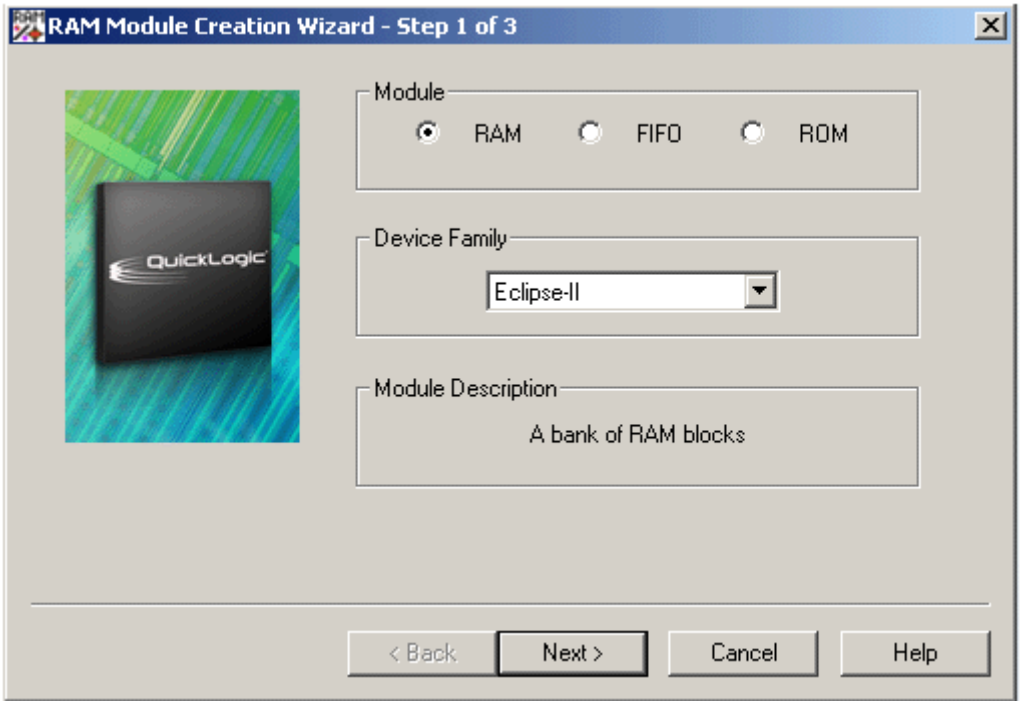
18.2 Creating RAM, FIFO and ROM Modules for Pinnacle, Pre-Pinnacle and Eclipse Devices

18.2.1 Creating RAM Blocks

1. In the Module area of the Creation Wizard, select **RAM**.

© 2008 QuickLogic Corporation

2. In the Device Family drop-down list box, select the device family you want to work with.



3. Click **Next**.
4. To define the type of RAM blocks to create, specify the **Depth** and **Width** of the RAM block in the Size area of the Creation Wizard. **Table 18-1** lists the depths available in the Depth drop-down box.

Table 18-1: RAM Block Depths

Device Architecture	Synchronous/Asynchronous
.35 µm	64, 128, 256, 512, 1024*, 2048*, 4096*
.25 µm	128, 256, 512, 1024, 2048*, 4096*
.18 µm	128, 256, 512*, 1024*, 2048*, 4096*

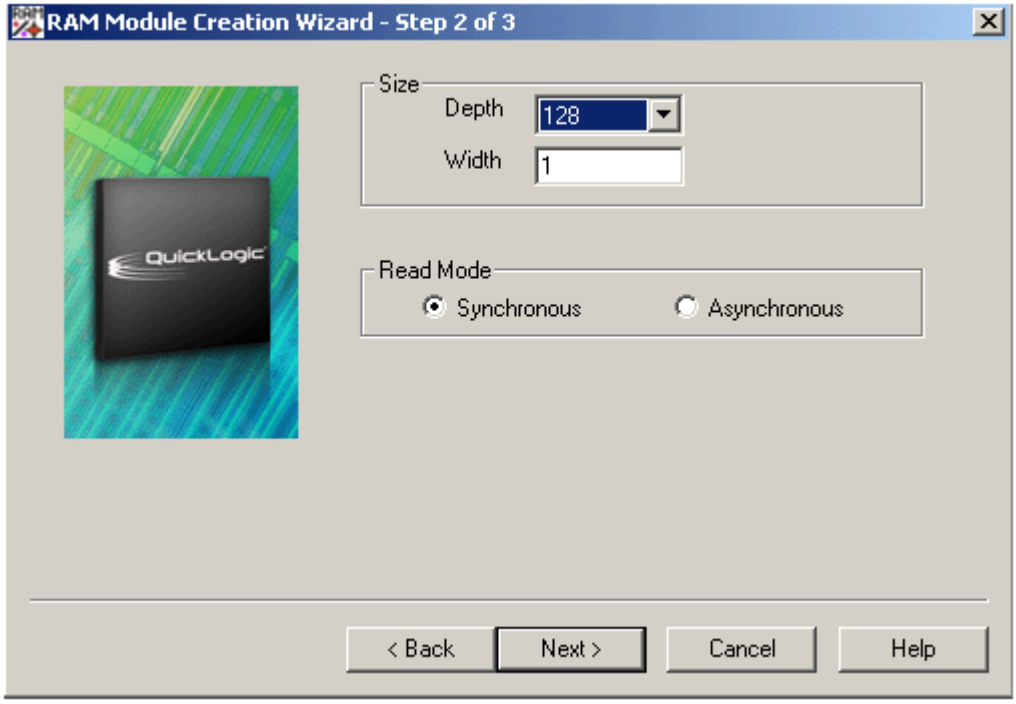
The Width of the RAM block to be entered must be in the range from `1` to `396` for QuickRAM devices and `1` to `864` for Eclipse devices.

NOTE: Any depth option with an asterisk (*) indicates that the RAM/ ROM/ FIFO Wizard will cascade multiple RAM blocks to generate the selected depth RAM block.

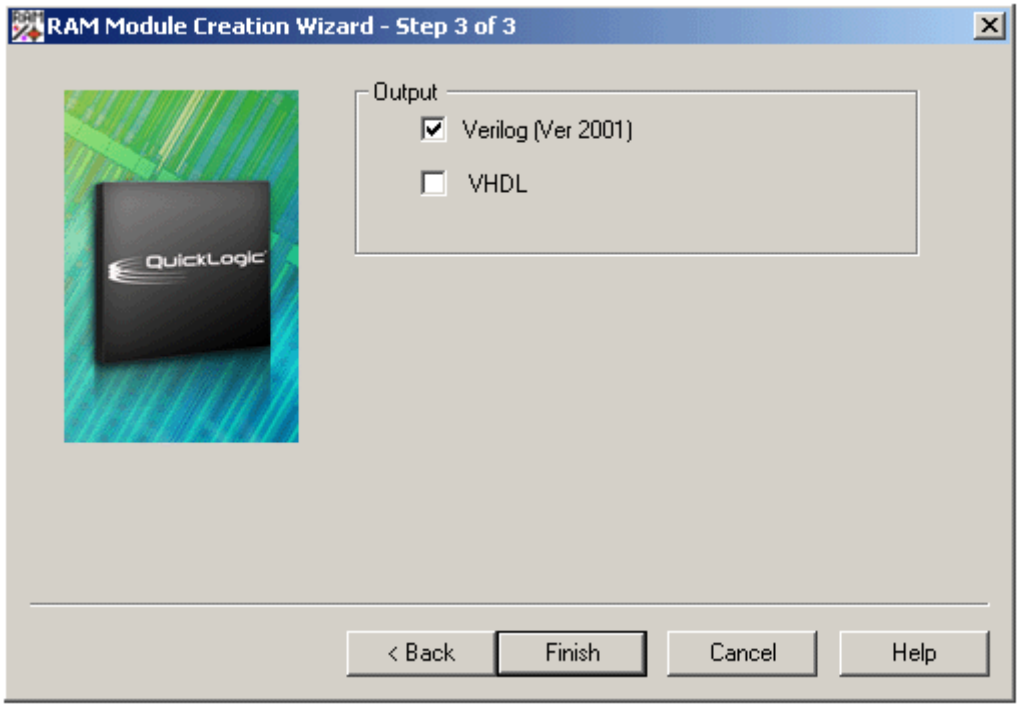
To select a QuickLogic RAM block read mode:

© 2008 QuickLogic Corporation

1. Select the appropriate option from the Read Mode area.



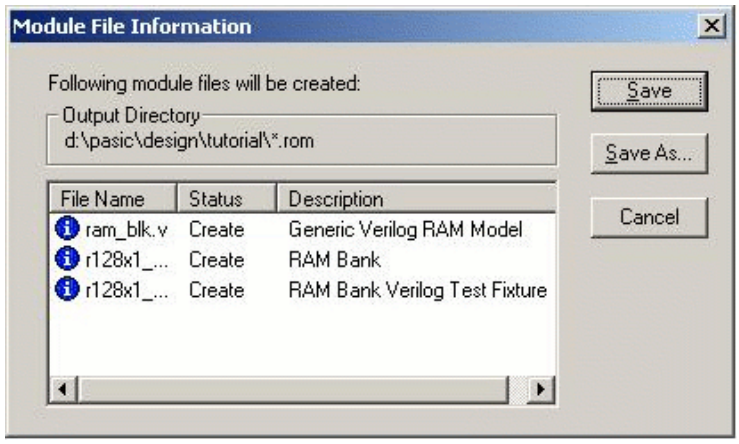
2. Click **Next**.
3. In the Output area of the Creation Wizard, choose the HDL Language (**Verilog** or **VHDL**) for the completed RAM block.



4. Click **Finish**.
The Module File Information box appears. The generated HDL and HDL Test Fixture files are listed in the Module File Information box.

© 2008 QuickLogic Corporation

5. Click **Save.**
   The RAM/ ROM/ FIFO Wizard saves the generated HDL files to the directory as shown in the Output Directory area.
   Or click **Save As** to select the directory to which the RAM/ ROM/ FIFO Wizard should save the generated HDL files.
   To cancel a current operation, click **Cancel**.
   To launch the SpDE Help files, click **Help**.



NOTE: SpDE Help files are available throughout the Creation Wizard.

18.2.1.1 RAM Design Files

Table 18-2 lists and describes the RAM files generated by the Creation Wizard.

Table 18-2: RAM Files

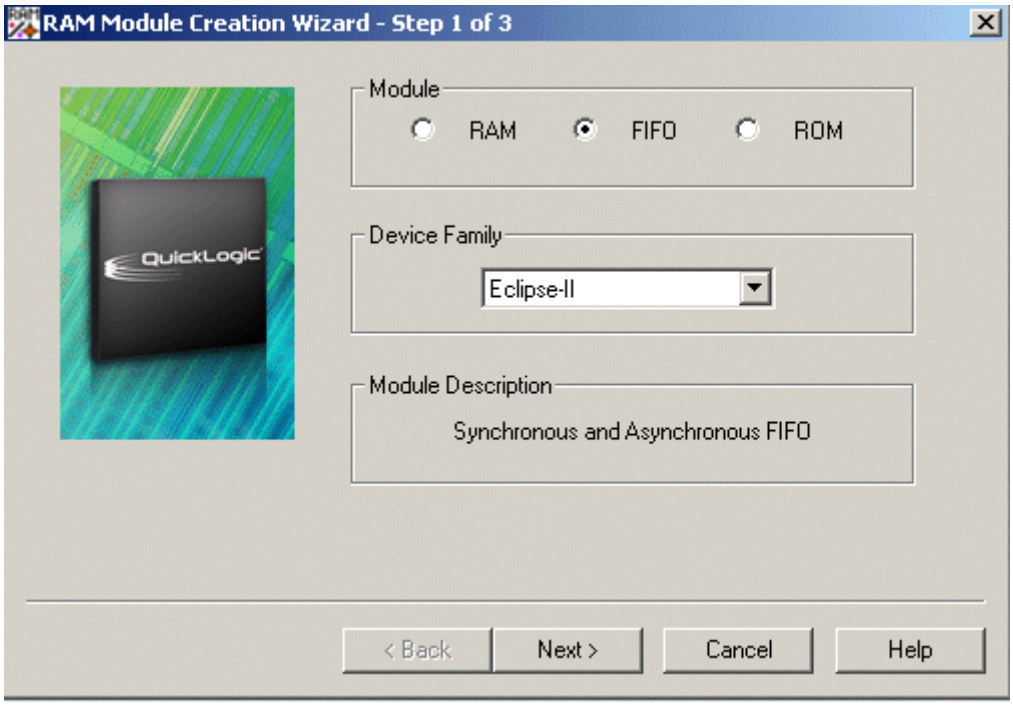
File	Description
`*.v`	The *.v Verilog file is the RAM Model design file. It conforms to specifications as selected in Creation Wizard. An x in the file name, for example, r256x1.v represents synchronous configuration. If asynchronous configuration was selected, the x in file name is replaced by an a, (r256a1.v).
`*.vhd`	The *.vhd VHDL file is the RAM Model design file. It conforms to specifications as selected in Creation Wizard. An x in the file name, for example, r256x1.vhd represents synchronous configuration. If asynchronous configuration was selected, the x in file name is replaced by an a, (r256a1.vhd).
`*.tf`	The *.tf Verilog Test Fixture file provides sample stimulus for the top-level design created by Creation Wizard, and for some of the sub-components.
`*.tb`	The *.tb VHDL Test Bench file provides sample stimulus for the top-level design created by Creation Wizard, and for some of the sub-components.

18.2.2 Creating FIFO Modules

1. In the Module area of the Creation Wizard, select the **FIFO** radio button.

© 2008 QuickLogic Corporation

2. In the Device Family drop-down list box, select the device family you want to work with.



3. Click **Next**.
4. To define the type of FIFO blocks to create, specify the Depth and Width of the FIFO block in the Size area of the Creation Wizard. The FIFO can be configured in synchronous or asynchronous mode, and other dependent options can be selected accordingly. The depths available in the drop-down menu for the FIFO options are listed in **Table 18-3**.

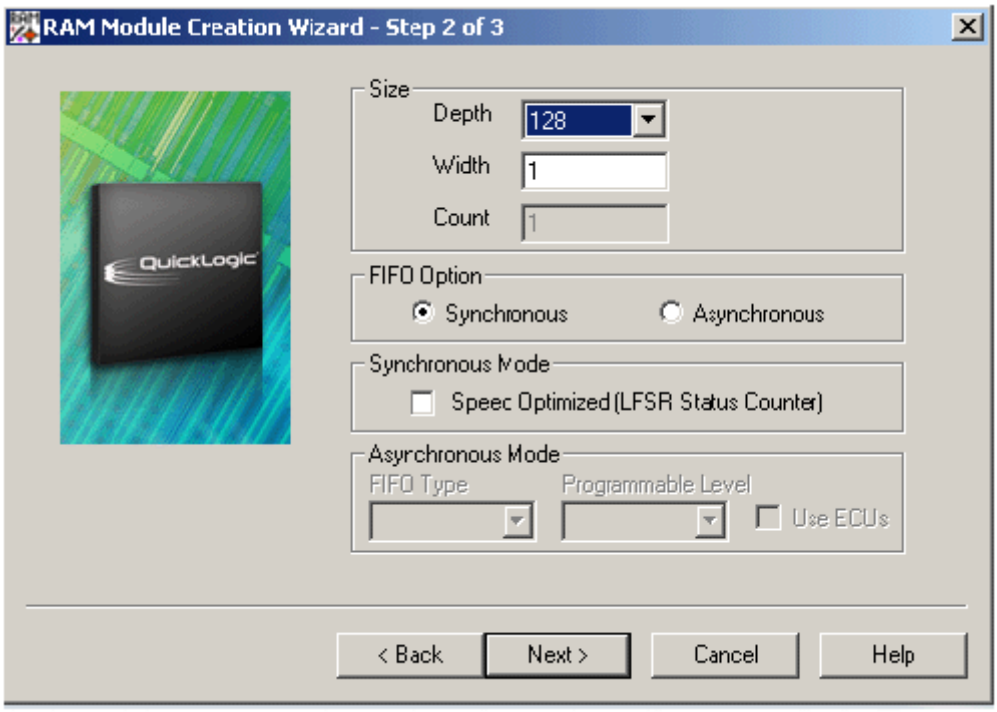
Table 18-3: FIFO Block Depths

Device Architecture	Synchronous	Asynchronous
.35 µm	64, 128, 256, 512	32, 64, 128, 256, 512, 1024
.25 µm	128, 256, 512, 1024	32, 64, 128, 256, 512, 1024
.18 µm	128, 256	32, 64, 128, 256

NOTE: Add 4,8,16 for Asynchronous FIFO with FIFO type *Normal.*

© 2008 QuickLogic Corporation

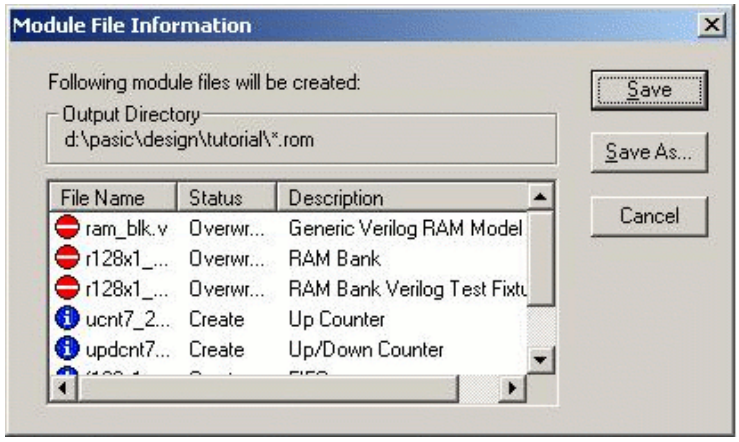
The width of the FIFO block to be entered must be in the range from `1` to `396` for QuickRAM devices and from `1` to `864` for Eclipse devices.



5. To select specific options for Synchronous and Asynchronous mode:
   - For Synchronous mode, the **Speed Optimized** option can be selected for better timing implementation.
   - For Asynchronous mode:
     - Select the FIFO type as **Normal**, **Programmable** or **Gas Gauge**
     - In case of Programmable FIFO type, the default programmable level **5** is applicable.
     - If the architecture for the selected device supports ECU blocks, the same can be selected for usage. By default, the option is not selected.
       Also, in Normal FIFO type, the ECUs are not used. Using ECUs gives an area optimized Async FIFO solution.
6. Click **Next**.
7. In the Output area of the Creation Wizard, choose the HDL Language (Verilog or VHDL) for the completed FIFO block.

© 2008 QuickLogic Corporation

8. Click **Finish**.
   The Module File Information box opens.



9. In the Module File Information box, the generated HDL and HDL Test Fixture/ Test Bench files are listed.
   Click **Save** and the Wizard will save the generated HDL files to the directory as shown in the Output Directory area.
   Or click **Save As** to select the directory to which the Wizard should save the generated HDL files.
   To cancel a current operation, click **Cancel**.
   For Asynchronous FIFO option, even the simulation scripts for ModelSim and Active HDL tools are dumped.

   NOTE: SpDE Help files are available throughout the Creation Wizard. To launch the SpDE Help files, click **Help**.

18.2.2.1 FIFO Design Files

Table 18-4 lists and describes RAM and FIFO files generated by the Creation Wizard.

Table 18-4: RAM and FIFO Files

File	Description
`*.v`	The `*.v` Verilog files are the RAM, FIFO, and Counter design files. It conforms to specifications as selected in Creation Wizard. For the creation of a FIFO, the Creation Wizard must generate a FIFO, RAM, and Counter (used to generate read and write addresses for FIFO) files based on the selected specifications. The FIFO file created by the Creation Wizard is saved in the following example format: `f256s9.v`.

Table 18-4: RAM and FIFO Files (Continued)

File	Description
`*.vhd`	The *.vhd VHDL files are the RAM, FIFO, and Counter design files. It conforms to specifications as selected in Creation Wizard. For the creation of a FIFO, the Creation Wizard must generate a FIFO, RAM, and Counter (used to generate read and write addresses for FIFO) files based on the selected specifications. The FIFO file created by the Creation Wizard is saved in the following example format: `f256s9.vhd`.
`*.tf`	The *.tf Verilog Test Fixture file provides sample stimulus for the top-level design created by the Creation Wizard, and for some of the sub-components. For the creation of a FIFO, the Creation Wizard generates FIFO and RAM test fixture files. The FIFO Test Fixture file created by the Creation Wizard is saved in the following example format: `f256s9.tf`.
`*.tb`	The *.tb VHDL Test Bench file provides sample stimulus for the top-level design created by the Creation Wizard, and for some of the sub-components. For the creation of a FIFO, the Creation Wizard generates FIFO and RAM test bench files. The FIFO Test Bench file created by the Creation Wizard is saved in the following example format: `f256s9.tb`.

NOTE: A detailed help on Asynchronous FIFO design functionality is available in **"Asynchronous FIFO Design Functionality" on page 236**.

18.2.3 Creating ROM Modules

To create a ROM from the Creation Wizard:

1. Create the Data Input file (`.rom` extension) – see the following `*.rom` file example (`rom1.rom`) in **Figure 18-1**. A `*.rom` file can be created using any text editor.

Figure 18-1: ROM Data Input File Example (rom1.rom)

```
rom=rom1
depth=256
width=8
asyncread=false
radix=hex
data
[0] = "00"
[1] = "01"
[2] = "02"
[3] = "03"
[4] = "04"
...
...
[254] = "FE"
[255] = "FF"
end
```


© 2008 QuickLogic Corporation

18.2.3.1 ROM Data Input File

Table 18-5 describes the `.rom` data input file keywords and settings.

Table 18-5: rom Data Input Files

Keyword/Setting	Description
rom	The text following the *rom* keyword specifies the name of the Verilog or VHDL file that will be generated.
depth and width	The numbers following *depth* and *width* specify the depth and width of ROM that will be created. These numbers should always be specified in decimal format.
asyncread	*Asyncread* is set to true for asynchronous read capabilities. Setting it to *false* will result in synchronous read capabilities.
radix	Set *radix* to either *binary* or *hex* to specify the format of data in the `.rom` file.
data	The keyword *data* tells the ROM generator that the proceeding lines specify the ROM data.
end	The keyword *end* after all the ROM data tells the compiler to stop reading ROM data.

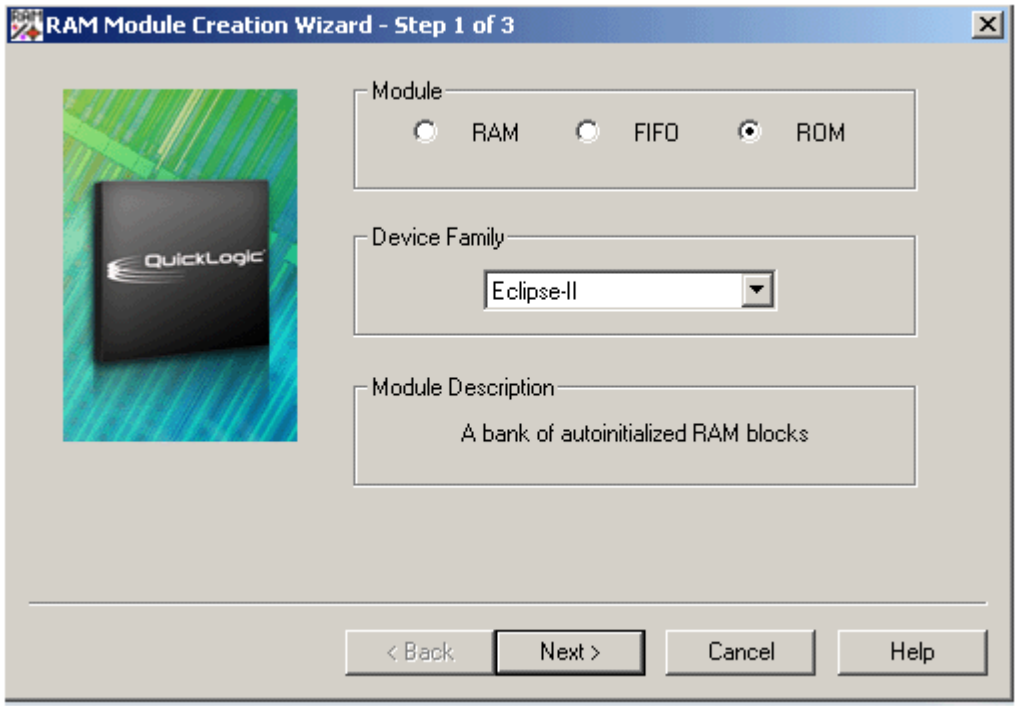
The format for the ROM data is the address (in decimal format), followed by actual data (written in *radix* specified by the *radix* attribute). Since the address is explicitly declared, you do not need to keep the data in ascending order. The values written to each address can contain more bits than the *width* specified in the *width* parameter with the truncation of extra bits. If the values written to each address contain fewer bits than that as specified in width parameter, an error will occur.

NOTE: All addresses must be explicitly defined. For instance, in the previous example, the depth was 256, therefore addresses 0 to 255 are explicitly defined.

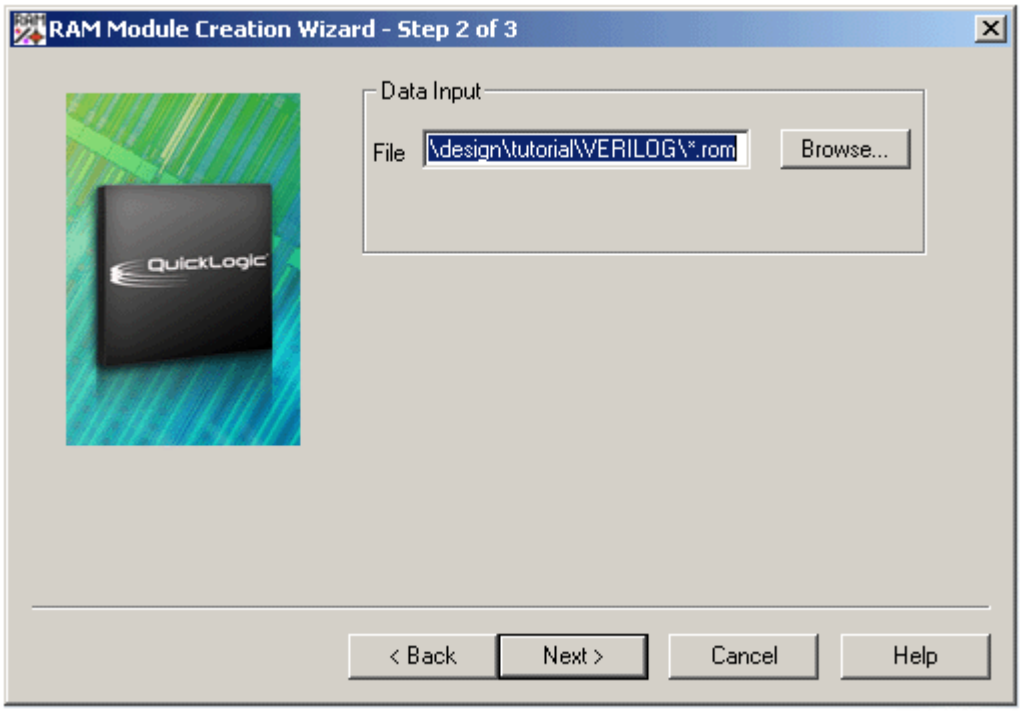
To create a ROM module:

1. In the Module area of the Creation Wizard, select **ROM**.

2. In the Device Family drop-down list box, select the device family you want to work with.



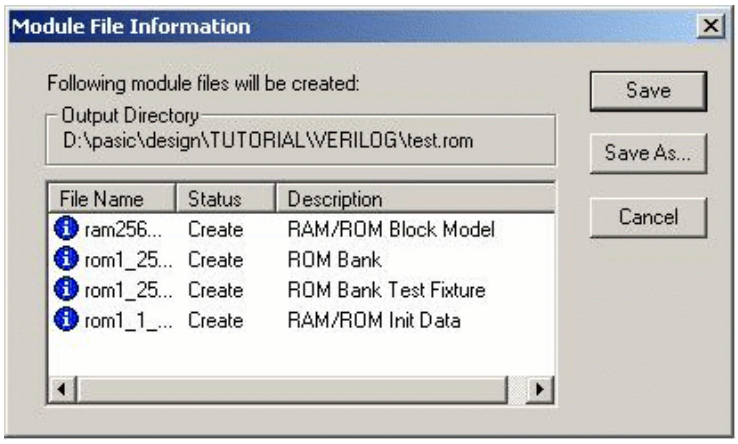
3. Click **Next**.
4. In the Data Input area of the Creation Wizard, type in the directory in which the `*.rom` file is saved, or click **Browse** to select or change directories in which the `*.rom` file is saved.



5. In the Output area of the Creation Wizard, choose the HDL Language (**Verilog** or **VHDL**) for the completed RAM block.

© 2008 QuickLogic Corporation

6. Click **Finish**.
   The Module File Information box opens.



7. In the Module File Information box, the generated HDL and HDL Test Fixture/ Test Bench files are listed.
   Click **Save** and the Wizard will save the generated HDL files to the directory specified in Output Directory area.
   Or click **Save As** to select the directory to which the Wizard should save the generated HDL files.
   To cancel the current operation, click **Cancel**.

   NOTE: SpDE Help files are available throughout the Creation Wizard. To launch the SpDE Help files, click **Help**.

18.2.3.2 ROM Design Files

Table 18-6 lists and describes RAM and FIFO files generated by the **Creation Wizard**.

Table 18-6: RAM and FIFO Files

File	Description
`*.v`	The `*.v` Verilog files are the RAM and ROM design files. It conforms to specifications as selected in Creation Wizard. For the creation of ROM, the Creation Wizard must generate a ROM and RAM files based on the selected specifications. The ROM file created by the Creation Wizard is saved under the user specified ROM filename with `.v` extension (e.g., `rom1.v`).
`*.vhd`	The `*.vhd` VHDL file are the RAM and ROM design files. It conforms to specifications as selected in Creation Wizard. For the creation of ROM, the Creation Wizard must generate a ROM and RAM files based on the selected specifications. The ROM file created by the Creation Wizard is saved under the user specified ROM filename with `.vhd` extension (e.g., `rom1.vhd`).

Table 18-6: RAM and FIFO Files

File	Description
*.tf	The *.tf Verilog Test Fixture file provides sample stimulus for the top-level design created by Creation Wizard, and for some of the sub-components. For the creation of ROM, the Creation Wizard generates a ROM test fixture file. The ROM Text Fixture file created by the Creation Wizard is saved under user specified ROM filename with .tf extension (e.g., rom1.tf).
*.tb	The *.tb VHDL Test Bench file provides sample stimulus for the top-level design created by Creation Wizard, and for some of the sub-components. For the creation of ROM, the Creation Wizard generates a ROM test bench file. The ROM Test Bench file created by the Creation Wizard is saved under user specified ROM filename with .tb extension (e.g., rom1.tb).
*.mem	The *.mem ROM initialization file is used by the simulator to load values into the RAM for simulation purposes. The .mem files are used by the tools to create the Intel Hex files necessary for writing the ROM information to the actual devices. The .mem ROM initialization file created by the Creation Wizard is saved in the format rom1_1.mem, with rom1 being the user-specified ROM filename.

NOTE: If you chose a configuration width and depth that uses an extended depth RAM, there will be other .v and .tf files where those modules are instantiated. However, the top-level file will always have a filename that corresponds to the width and depth of the user specifications. For example, if you defined a synchronous RAM with a width of 10 and a depth of 2048 the resulting top-level file would be r2048x10.v.

18.2.3.3 Using the RAM/ROM/FIFO Components in HDL Designs

Once the HDL components are created, these components need to be instantiated into Verilog or VHDL source codes.

For an example on how to instantiate macros in HDL source codes, refer to *QuickNote #58* posted at www.quicklogic.com.

18.2.4 Asynchronous FIFO Design Functionality

Asynchronous FIFOs are useful in a wide range of complex digital systems, where transferring or buffering data between two asynchronous clock domains is required. Data is stored using one clock domain and retrieved using another clock domain. Status flags generated by logic in each clock domain are implemented to provide external control logic with an indication of the amount of data currently stored in the FIFO.

18.2.4.1 QuickLogic Asynchronous FIFO Features

The parameterized asynchronous FIFO core implemented into the FIFO wizard enables the choice of status flag generation to suit the varying needs of system designers.

The following is a list of the available features supported in the FIFO wizard for asynchronous FIFO generation:

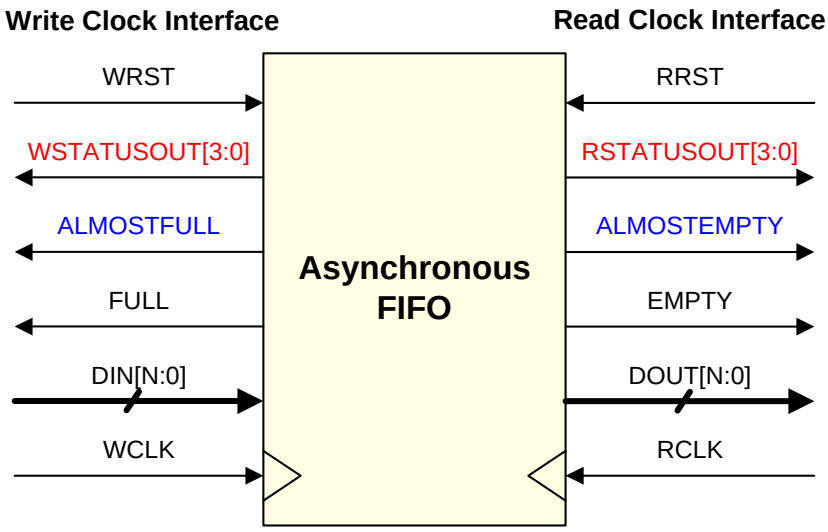
- Integration into SpDE for automatic file generation of core design and simulation files
- Device and architecture specific optimization of FIFO core
- Parameterized word width and data depth

© 2008 QuickLogic Corporation

- *Normal*, *Programmable*, and *Gas Gauge* options for FIFO type
- ECU utilization for area optimized designs

Figure 18-2 illustrates the write and read clock interfaces of the asynchronous FIFO.

Figure 18-2: Asynchronous FIFO Read/Write Clock Interfaces



18.2.4.2 Asynchronous FIFO Overview

The selection of the required word width and data depth of the asynchronous FIFO and the specification of the FIFO type desired is done through the FIFO wizard user interface. **Table 18-7** illustrates the available asynchronous FIFO options and the associated FIFO status flag outputs. The three sets of status flag outputs provide a unique view of the state of the FIFO and each could be potentially useful to the designed operation of the FIFO controller.

Table 18-7: Asynchronous FIFO Options

FIFO Type	FIFO Status Flag Outputs	Description
Normal	EMPTY FULL	Registered outputs which indicate whether the FIFO is full or empty
Programmable	EMPTY FULL *ALLMOSTEMPTY* *ALMOSTFULL*	Additional status flags - registered outputs which indicate whether the FIFO is within a fixed threshold of becoming full or empty
Gas Gauge	EMPTY FULL *RSTATUSOUT[3:0]* *WSTATUSOUT[3:0]*	Additional status flags - registered outputs which provide a specified range of the number of data words currently stored in the FIFO

© 2008 QuickLogic Corporation

Table 18-8 lists the asynchronous FIFO user interface signals. A short description of each signal and its port characteristics are provided.

Table 18-8: Asynchronous FIFO Interface Signal Description

Signal	Signal Type	Description
RRST	Input	Read clock domain reset
WRST	Input	Write clock domain reset
RCLK	Input	Read clock
WCLK	Input	Write clock
PUSH	Input	Data write request
DIN[N:0]	Input	Data input
POP	Input	Data read request
DOUT[N:0]	Output	Data output
EMPTY	Output	FIFO empty signal (synchronous to RCLK)
ALMOSTEMPTY	Output	FIFO almost empty signal – 5 or fewer data words in FIFO available to be read (synchronous to RCLK, one clock frequency)
RSTATUSOUT[3:0]	Output	RSTATUSOUT[3]: asserted high when FIFO is ¾ full – full RSTATUSOUT[2]: asserted high when FIFO is ½ full – [¾ full - 1] RSTATUSOUT[1]: asserted high when FIFO is ¼ full – [½ full - 1] RSTATUSOUT[0]: asserted high when FIFO is empty – [¼ full - 1] (synchronous to RCLK, one clock frequency)
FULL	Output	FIFO full signal (synchronous to WCLK)
ALMOSTFULL	Output	FIFO almost full signal - 5 or fewer locations available in FIFO to be written (synchronous to WCLK, one clock frequency)
WSTATUSOUT[3:0]	Output	WSTATUSOUT[3]: asserted high when FIFO is ¾ full – full WSTATUSOUT[2]: asserted high when FIFO is ½ full – [¾ full - 1] WSTATUSOUT[1]: asserted high when FIFO is ¼ full – [½ full - 1] WSTATUSOUT[0]: asserted high when FIFO is empty – [¼ full - 1] (synchronous to WCLK, one clock frequency)

18.2.4.3 Functional Description and Timing Waveforms

Figure 18-3 illustrates single read and write operations to the FIFO. A write operation is executed when the *push* input is asserted (HIGH) and the *full* flag is inactive (LOW) at the rising edge of *wclk*. The first data word written to the FIFO will be available at the output on the next rising edge of *rclk*. Subsequent read operations will pop data out of the FIFO sequentially.

© 2008 QuickLogic Corporation

A read operation is executed when the *pop* input is asserted (HIGH) and the *empty* flag is inactive (LOW) at the rising edge of *rclk*. The next data word in the FIFO will be available at the output on the next rising edge of *rclk*. Write or read requests that cause overflow or underflow conditions are invalid and will be ignored.

Figure 18-3: Single Read/Write Operations

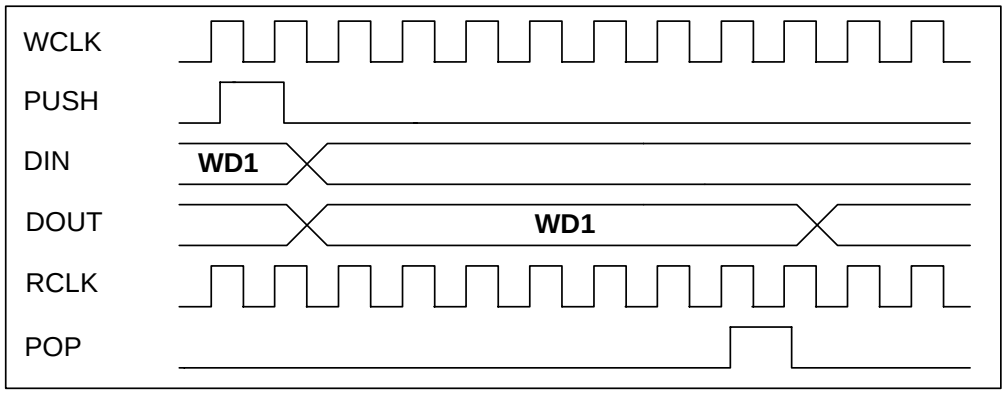


Figure 18-4 illustrates consecutive write operations to fill the FIFO and the associated status flag timing in the write clock domain. The *push* input is asserted (HIGH) until the *full* flag becomes asserted (HIGH) at the tenth rising edge of *wclk*.

WD8 is the last data word written into the FIFO.

The *almostfull* status flag is asserted (HIGH) when five or fewer locations are available in the FIFO to be written. This occurs after the third data word is written to the eight-deep FIFO. There is one clock cycle of latency associated with the generation of this flag in the write clock domain.

The *wstatusout[3:0]* status flag specifies a specific range of the occupancy of the FIFO. The FIFO is empty to [¼ full - 1] when wstatusout[3:0] takes on the value of *1h*; ¼ full to [½ full - 1] at *2h*; ½ full to [¾ full - 1] at *4h*; ¾ full to full at *8h*. There is one clock cycle of latency associated with the generation of these flags in the write clock domain.

© 2008 QuickLogic Corporation

The deassertion of the *empty* flag as seen below is delayed due to the latency introduced by registers used to synchronize the read pointer address to *wclk*. For the same reason, the *almostfull* and *wstatusout[3:0]* status flags will not respond immediately to changes in state caused by read operations. There may be up to 3 clock cycles of latency for read operations (read clock domain) to be updated in the write clock domain.

Figure 18-4: Write Timing Waveforms

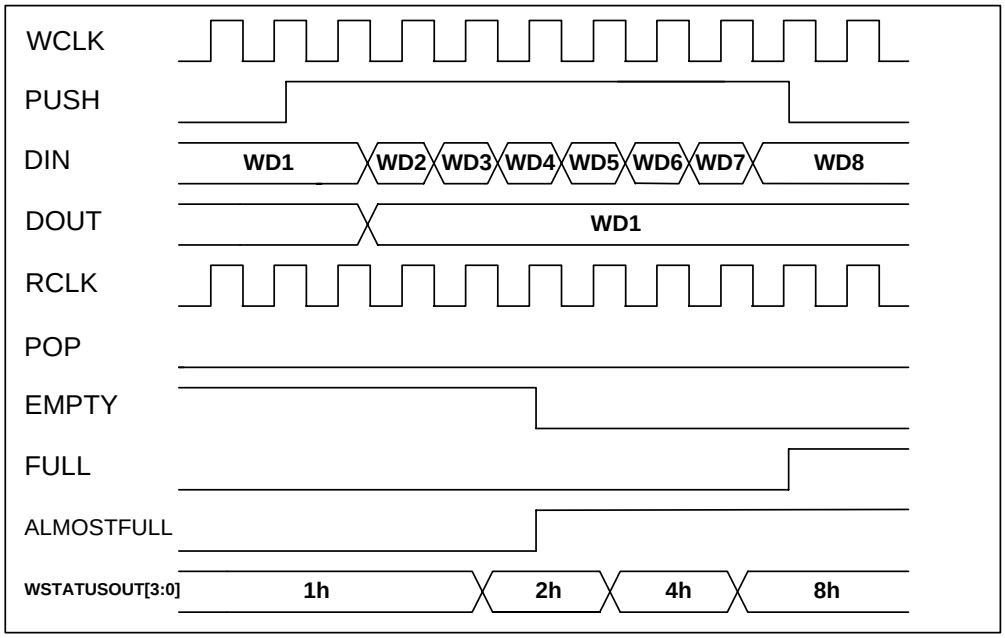


Figure 18-5 illustrates consecutive read operations to empty the FIFO and the associated status flag timing in the read clock domain. The *pop* input is asserted (HIGH) until the *empty* flag becomes asserted (HIGH) at the tenth rising edge of *rclk*.

WD8 is the last data word read from the FIFO.

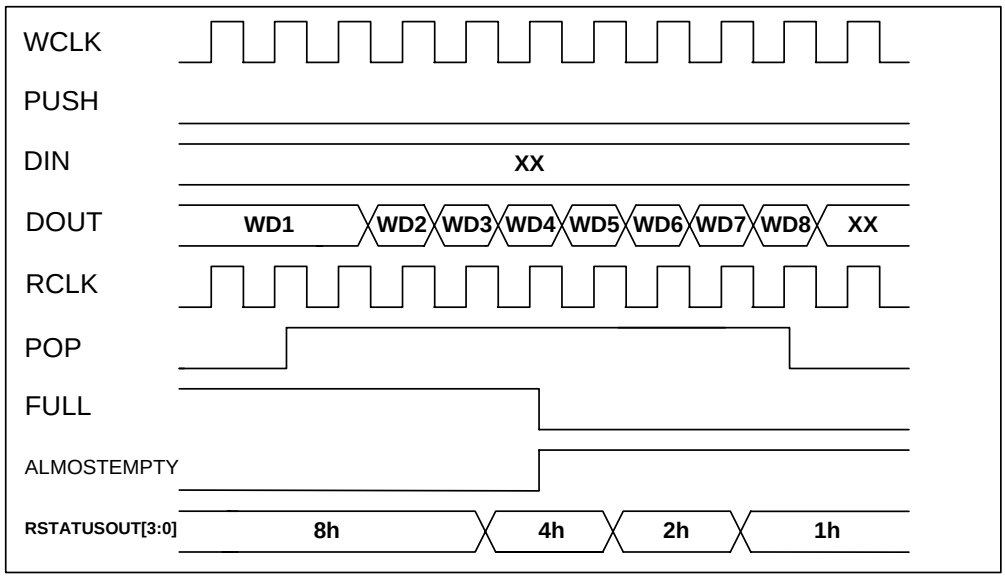
The *almostempty* status flag is asserted (HIGH) when five or fewer locations are available in the FIFO to be read. This occurs after the third data word is read from the eight-deep FIFO. There is one clock cycle of latency associated with the generation of this flag in the read clock domain.

The *rstatusout[3:0]* status flag specifies a specific range of the occupancy of the FIFO. The FIFO is empty to [¼ full -1] when rstatusout[3:0] takes on the value *1h*; ¼ full to [½ full - 1] at *2h*; ½ full to [¾ full - 1] at *4h*; ¾ full to full at *8h*. There is one clock cycle of latency associated with the generation of these flagsin the read clock domain.

© 2008 QuickLogic Corporation

The deassertion of the *full* flag as seen below is delayed due to the latency introduced by registers used to synchronize the write pointer address to *rclk*. For the same reason, the *almostempty* and *rstatusout[3:0]* status flags will not respond immediately to changes in state caused by write operations. There may be up to 3 clock cycles of latency for write operations (write clock domain) to be updated in the read clock domain.

Figure 18-5: Read Timing Waveforms



18.2.4.4 Utilization and Performance Benchmarks

Table 18-9 illustrates several commonly implemented asynchronous FIFO configurations implemented in the programmable fabric of an Eclipse II device (QL8325-8ps484). The respective logic cell utilization and performance information are shown for easy reference. ECU utilization, where available, will preserve additional logic cell resources.

Table 18-9: Asynchronous FIFO Implementation in Programmable Logic (Eclipse II)

Async FIFO Option	Logic Cell Utilization	RAM	ECU	Performance
Async 32x32 FIFO (Normal)	29	2	0	207 MHz
Async 128x32 FIFO (Normal)	41	2	0	176 MHz
Async 32x32 FIFO (Programmable)	62 45	2	0	181 MHz/135 MHz
Async 128x32 FIFO (Programmable)	86 62	2	0 2	170 MHz/136 MHz
Async 32x32 FIFO (Gas Gauge)	57 47	2	0 2	189 MHz/149 MHz
Async 128x32 FIFO (Gas Gauge)	76 64	2	0 2	172 MHz/141 MHz

© 2008 QuickLogic Corporation

18.2.4.5 Asynchronous FIFO Core Package

The asynchronous FIFO core package contains the design files in Verilog and VHDL shown in **Table 18-10**. The order of synthesis and a short description of each core design file are provided.

Table 18-10: Core Design Files

Order of Synthesis	File Name	Description
1	macros	Contains QuickLogic macro definitions
2	ecu	Contains QuickLogic ecu definitions
3	afifopackage	Contains macro definitions specific to FIFO flag and address generation
4	ram_blk	Contains RAM block module generation code
5	rstatus_t*x*	Design file (contains logic for FIFO flag generation in read clock domain)
6	wstatus_t*x*	Design file (contains logic for FIFO flag generation in write clock domain)
7	asyncfifo_t*x*	Design file (parameterized async FIFO core)
8	af*xxxx*_t*x*	Top-level design file

Table 18-11 lists the simulation files provided with each asynchronous FIFO configuration. The design testbench instantiates asyncfifo_tx with the generic parameters specified in asyncfifo_tb_pkg. Parameters in asyncfifo_tb_pkg may be modified if desired.

Table 18-11: Simulation Files

File Name	Description
asyncfifo_tb_pkg	Testbench package file (defines constant values used in asyncfifo_t*x*_tb)
asyncfifo_t*x*_tb	Design testbench (instantiates asyncfifo_t*x*)
asyncfifo_t*x*_aldec_pre.do asyncfifo_t*x*_modelsim_pre.do	Aldec and ModelSim macro files to automatically execute simulations

NOTE: Verilog simulators/synthesis tools must support Verilog 2001 to successfully perform functional simulation and gate-level synthesis.

© 2008 QuickLogic Corporation

18.2.4.6 Asynchronous FIFO Applications

Figure 18-6 and **Figure 18-7** illustrate several applications where an asynchronous FIFO would be useful as a solution for system designers using QuickLogic FPGAs and ESPs.

Figure 18-6: Interface between PCI and SDRAM

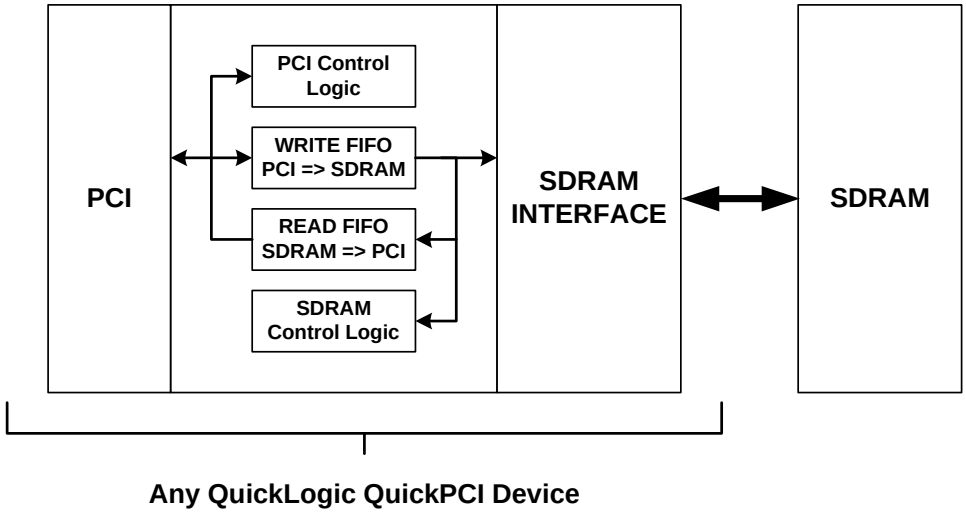
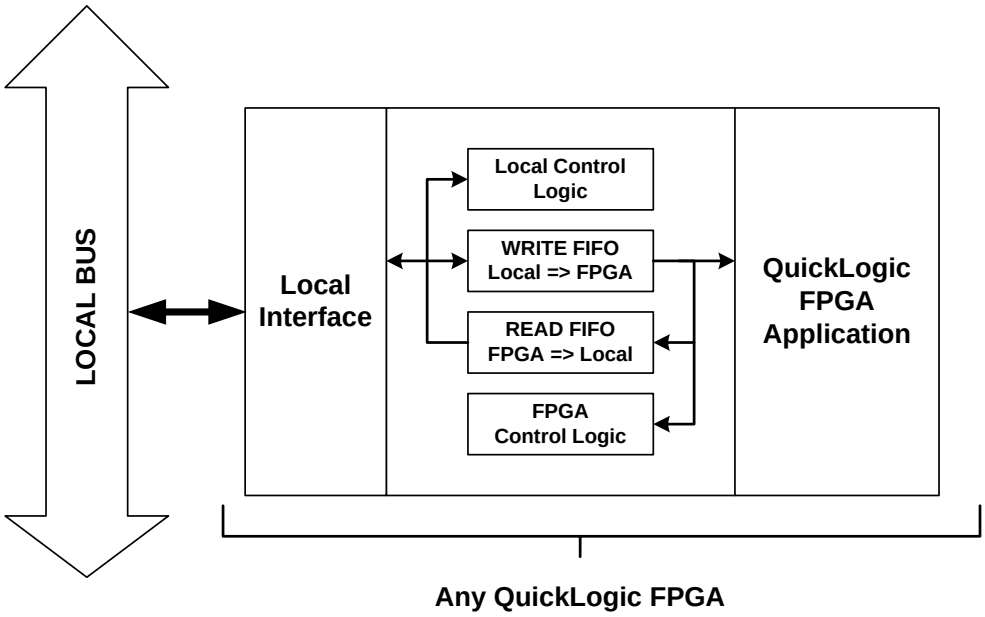


Figure 18-7: Interface between Local Interface and FPGA

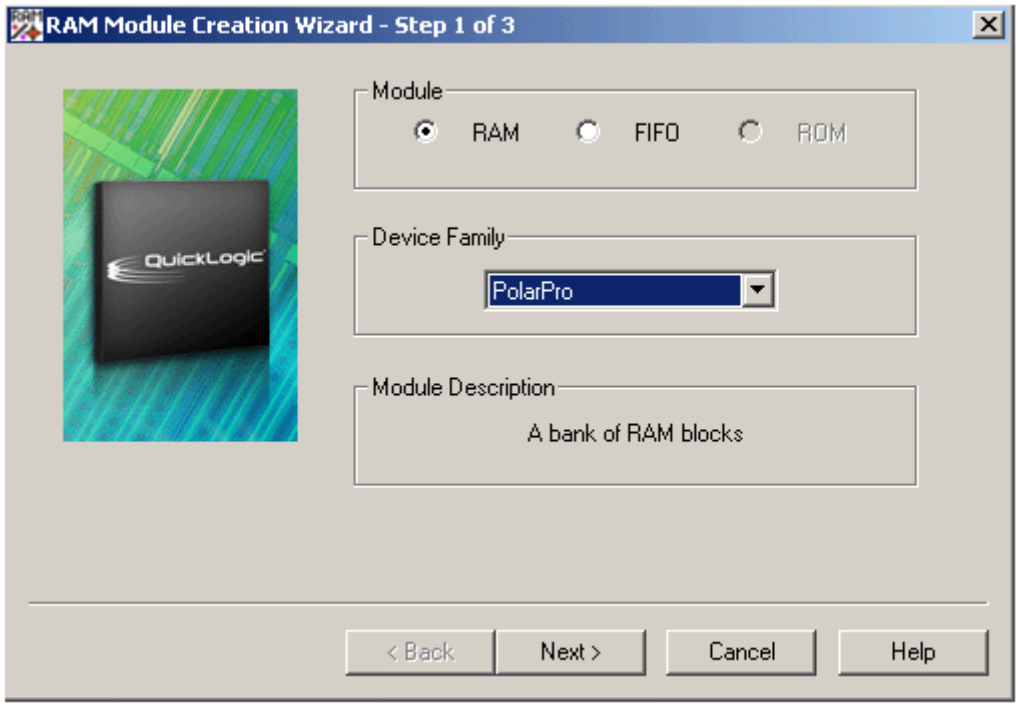


18.3 Creating RAM and FIFO Modules for PolarPro and PolarPro II Devices

NOTE: PolarPro devices do not have a ROM.

18.3.1 Creating RAM Blocks

1. In the Module area of the Creation Wizard, select **RAM.**
2. In the Device Family drop-down list box, select the **PolarPro** device family.



3. Click **Next**.
4. Select the appropriate part name.
5. To define the type of RAM blocks to create, specify the **Write Depth, Write Width, Read Depth** and **Read Width** of the RAM block in the Size area of the Creation Wizard. **Table 18-12** lists the valid depth and width combinations.

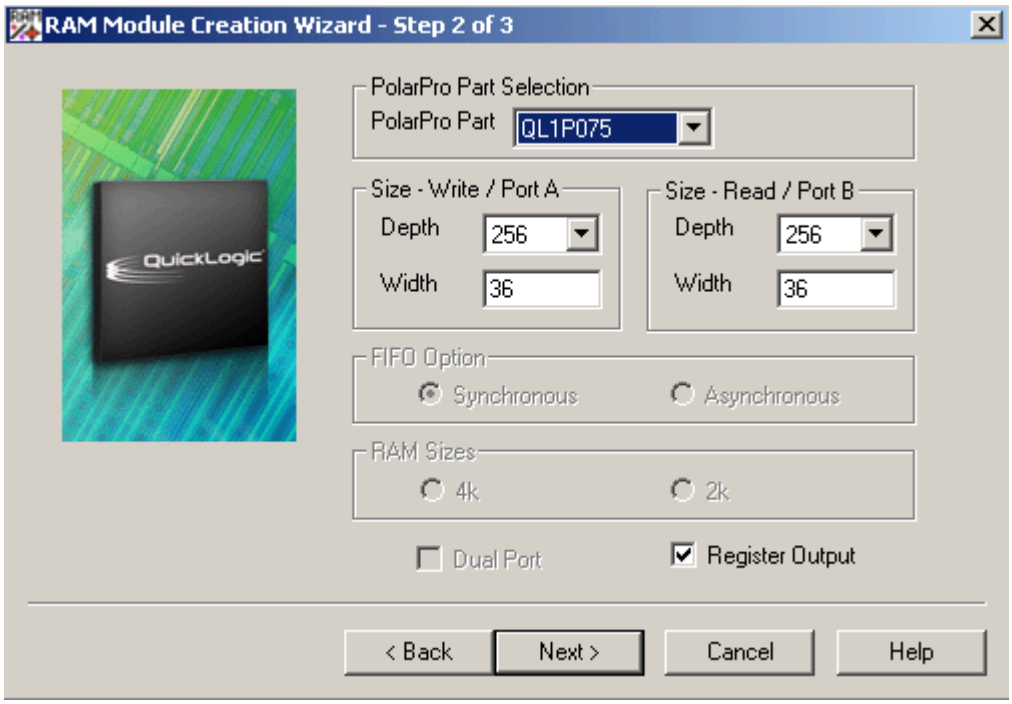
Table 18-12: PolarPro Depths and Widths

Depth	Width
256	1 to 36
512	1 to 18
1024	1 to 9

6. If you prefer to receive registered output, select **Register Output**.

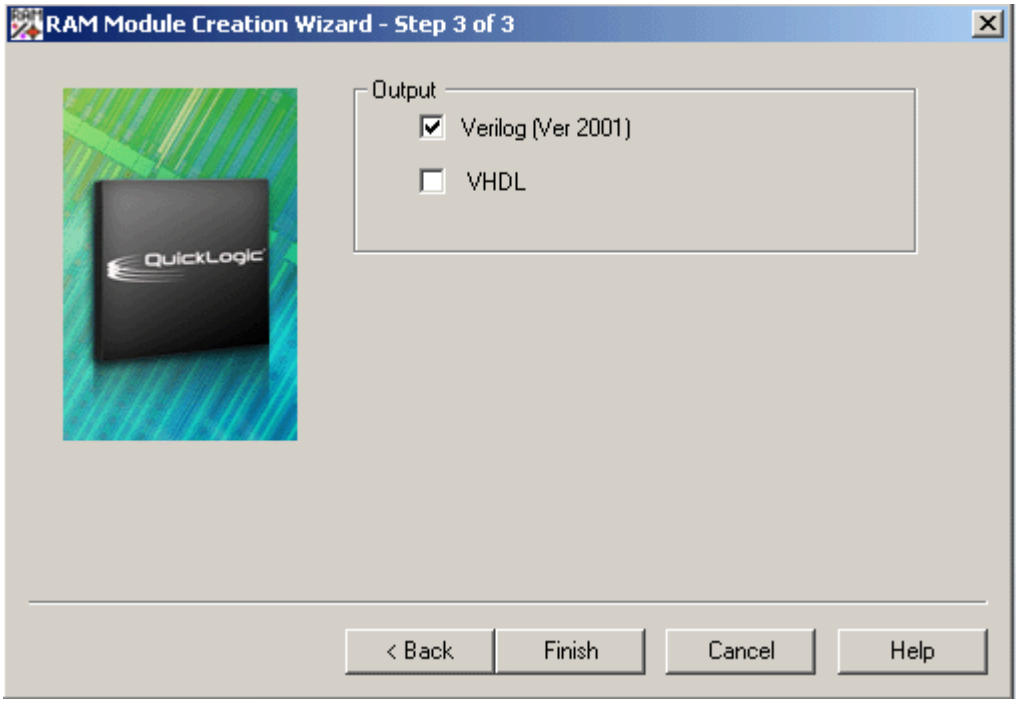
NOTE: If a write or read depth is selected and any one of the width values is entered, the other width is selected automatically. If write and read depths are related by a ratio of 2:1, then the higher width entered must be an even number. Similarly, if write and read depths are related by a ratio of 4:1, the higher width entered should be a multiple of four.

© 2008 QuickLogic Corporation



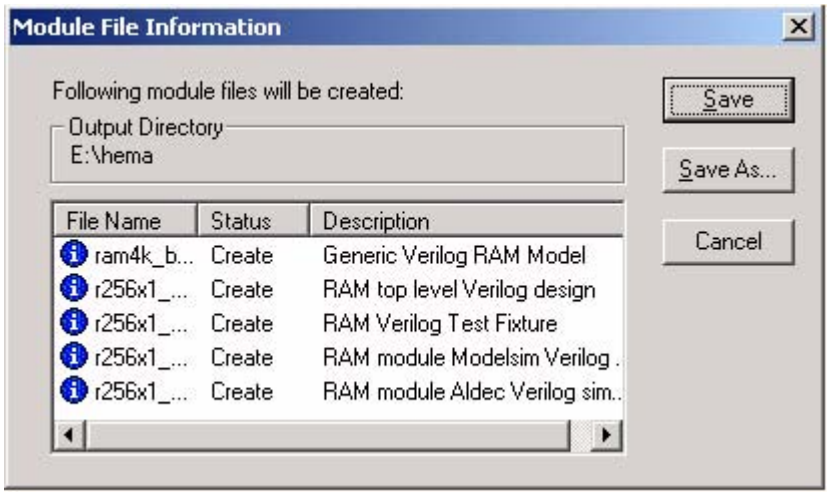
To select the output language:

1. Click **Next**.
2. In the Output area, select **VHDL** or **Verilog**.



To store the output files:

1. Click **Finish**.
   The Module File Information dialog box appears. The generated HDL and HDL Test Fixture files are listed in the Module File Information box.
2. Click **Save**.
   The RAM/ ROM/ FIFO Wizard saves the generated HDL files to the directory as shown in the Output Directory area.
   Or click **Save As** to select the directory to which the RAM/ ROM/ FIFO Wizard should save the generated HDL files.
   To cancel the current operation, click **Cancel**.



18.3.1.1 Design Files

Table 18-13 lists and describes the RAM files generated by the Creation Wizard.

Table 18-13: PolarPro RAM Files

File	Description
`*.v`	Verilog output file for the top level RAM module. The filename depends on the write/read widths and depths. For example, if write depth is 256, write width is 18, read depth is 512 and read width is 9, the file name is `r256x18_512x9.v`
`*.vhd`	VHDL output file for the top level module. For the above mentioned example, the module name would be `r256x18_512x9.vhd`.
`*.do`	There are four `.do` files. These files contain the script to be executed by Modelsim and ActiveHDL. The filenames for the above example would be as follows: `r256x18_512x9_Modelsim_Verilog_pre.do`, `r256x18_512x9_Aldec_Verilog_pre.do`, `r256x18_512x9_Modelsim_vhdl_pre.do` and `r256x18_512x9_Aldec_vhdl.do`.
`Ram4k_blk.v`	Generalized RAM module which contains the instantiation of the ram8k model. It is a Verilog file.

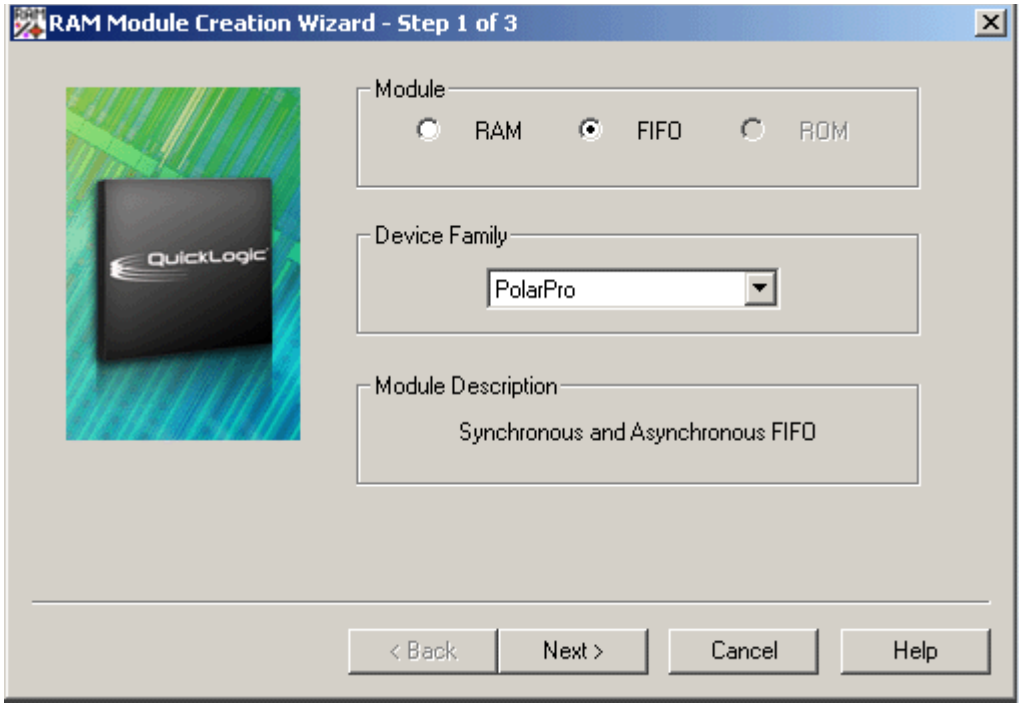
© 2008 QuickLogic Corporation

Table 18-13: PolarPro RAM Files (Continued)

File	Description
Ram4k_blk.vhd	Generalized RAM module which contains the instantiation of the ram8k model. It is a VHDL file.
*.tb	Testbench file for VHDL design. It provides stimulus for the top-level design file.
*.tf	Verilog testbench file. It provides stimulus for the top-level design file.

18.3.2 Creating FIFO Modules

1. In the Module area of the Creation Wizard, select **FIFO.**
2. In the Device Family drop-down list box, select the **PolarPro** device family.



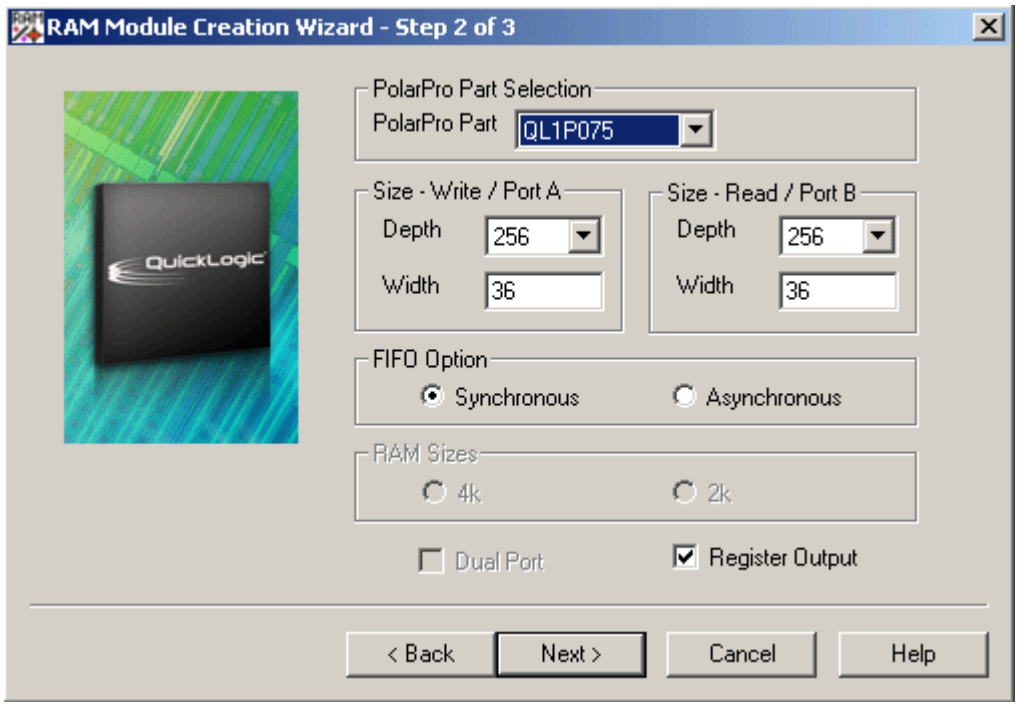
3. Click **Next.**
4. To define the type of FIFO blocks to create, specify the **Write Depth, Read Depth, Write Width** and **Read Width** of the FIFO block in the Size area of Creation Wizard. The FIFO can be configured in synchronous or asynchronous mode, and other dependent options can be selected accordingly.
   Table 18-14 lists the valid depth and width combinations.

Table 18-14: PolarPro Depths and Widths

Depth	Width
256	1 to 36
512	1 to 18
1024	1 to 9

5. If you prefer to receive registered output, select **Register Output**.

NOTE: If a write or read depth is selected and any one of the width values is entered, the other width is selected automatically. If write and read depths are related by a ratio of 2:1, then the higher width entered must be an even number. Similarly, if write and read depths are related by a ratio of 4:1, the higher width entered should be a multiple of four.

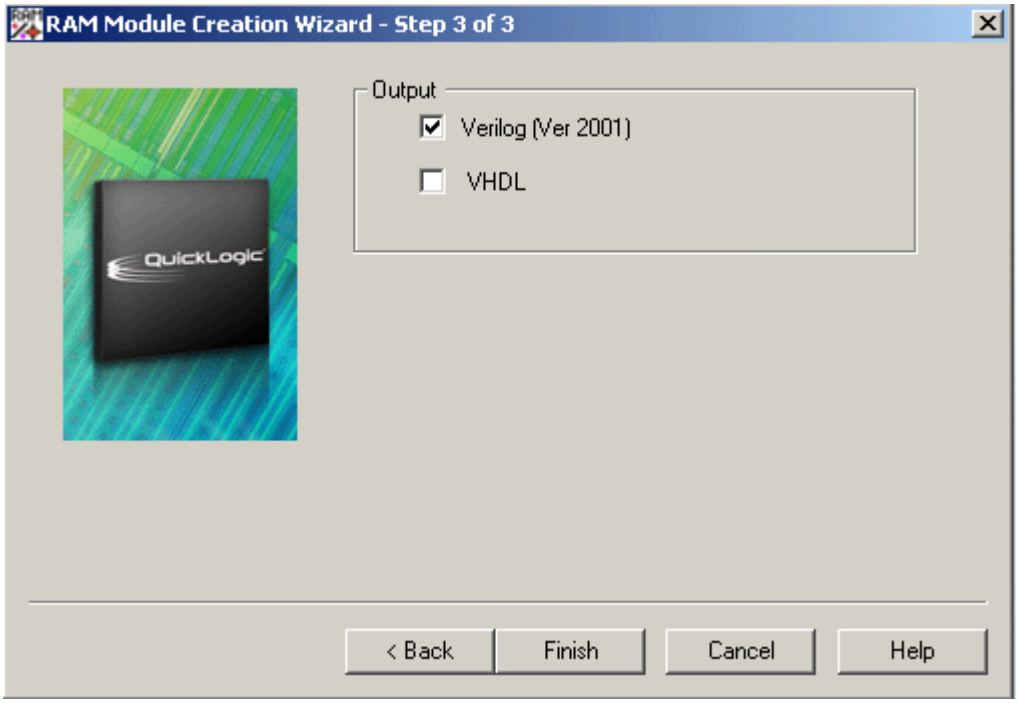


To choose the language:

1. Click **Next**.

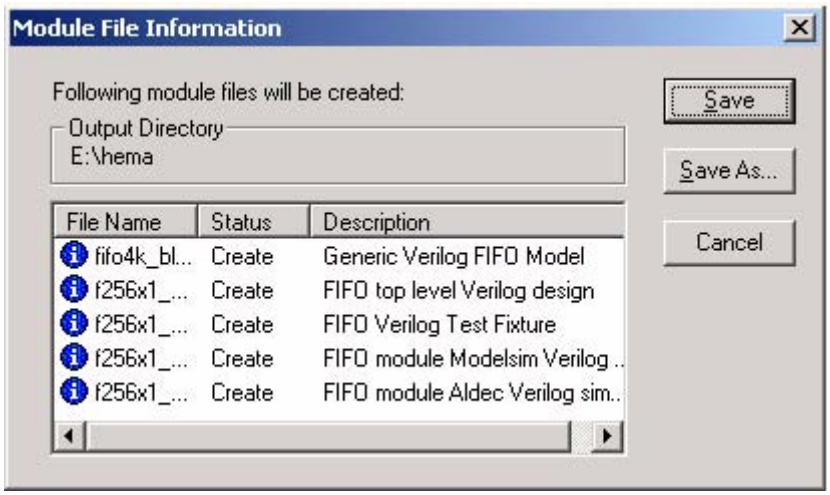
© 2008 QuickLogic Corporation

2. In the Output area, select **VHDL** or **Verilog**.



To store the output:

1. Click **Finish**.
   The Module File Information box appears. The generated HDL and HDL Test Fixture files are listed in the Module File Information dialog box.
2. Click **Save**.
   The RAM/ ROM/ FIFO Wizard saves the generated HDL files to the directory as shown in the Output Directory area.
   Or click **Save As** to select the directory to which the RAM/ ROM/ FIFO Wizard should save the generated HDL files.
   To cancel a current operation, click **Cancel**.



18.3.2.1 FIFO Design Files

Table 18-15 lists and describes the FIFO files generated by the Creation Wizard.

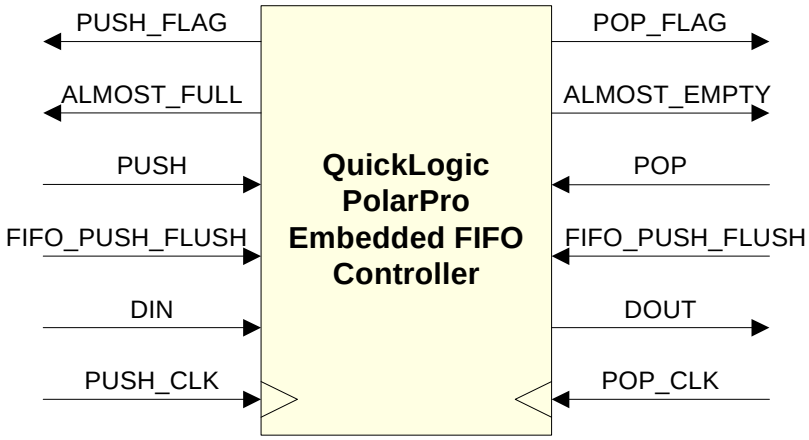
Table 18-15: PolarPro FIFO Files

File	Description
`*.v`	Verilog output file for the top level FIFO module. The filename depends on the write/read widths and depths. For example, if write depth is 256, write width is 18, read depth is 512 and read width is 9, the file name is `f256x18_512x9.v` for synchronous FIFO and `af256x18_512x9.v` for asynchronous FIFO.
`*.vhd`	VHDL output file for the top level FIFO. For the above mentioned example, the module name would be `f256x18_512x9.vhd` for synchronous FIFO and `af256x18_512x9.vhd` for asynchronous FIFO.
`*.do`	There are four `.do` files for synchronous FIFO and four for asynchronous FIFO. These files contain the scripts to be executed by the ActiveHDL and Modelsim simulators.
`Fifo4k_blk.v`	This is a Verilog file containing a generalized FIFO module. It instantiates the ram8k model.
`Fifo4k_blk.vhd`	This is a VHDL file containing a generalized FIFO module. It instantiates the ram8k model.
`*.tb`	Testbench file for a VHDL design. It provides stimulus for the top-level design file.
`*.tf`	Testbench file for a Verilog design. It provides stimulus for the top-level FIFO file.

18.3.2.2 FIFO Interface Signals

Figure 18-8 illustrates the read and write clock interfaces of the FIFO for **PolarPro** devices.

Figure 18-8: PolarPro FIFO Read/Write Clock Interfaces



© 2008 QuickLogic Corporation

18.3.2.3 FIFO Interface Signal Description

Table 18-16 lists the asynchronous FIFO user interface signals for PolarPro devices. A short description of each signal and its port characteristics are provided.

Table 18-16: PolarPro FIFO Signal Descriptions

Signal	Signal Type	Description
PUSH_CLK	Input	Write clock
PUSH	Input	Data write request
DIN	Input	Input data
FIFO_PUSH_FLUSH	Input	Signal to flush the data on the write side.
ALMOST_FULL	Output	This signal is asserted when the FIFO has room for only one or is FULL.
PUSH_FLAG	Output	FIFO push level indicator. The following table lists its values and descriptions. 0000 Full 0001 Empty 0010 Room for more than ½ 0011 Room for more than ¼ 1000 Room for 8 or more 1001 Room for 7 1010 Room for 6 1011 Room for 5 1100 Room for 4 1101 Room for 3 1110 Room for 2 1111 Room for 1 Others Reserved
POP_CLK	Input	Read clock
POP	Input	Signal to request read operation
DOUT	Output	Data output

Table 18-16: PolarPro FIFO Signal Descriptions (Continued)

Signal	Signal Type	Description
FIFO_POP_FLUSH		Signal to flush the data
ALMOST_EMPTY	Output	Almost empty indicator. This signal is asserted when the FIFO has one left or is EMPTY.
POP_FLAG	Output	FIFO pop level indicator. The following table lists its values and descriptions 0000 Empty 0001 1 entry in FIFO 0010 2 entries in FIFO 0011 3 entries in FIFO 0100 4 entries in FIFO 0101 5 entries in FIFO 0110 6 entries in FIFO 0111 7 entries in FIFO 1000 8 or more entries in FIFO 1101 ¼ or more full 1110 ½ or more full 1111 Full Others Reserved

The RAM resources supported by the ArcticLink devices are the same as the PolarPro devices. In the PolarPro II devices there are two types of RAM resources, the 2 K RAM block and the 4 K RAM block.

In case of FIFO configuration, the RAM resources supported by the PolarPro II devices are different than the PolarPro devices. The PolarPro II RAM has one input (DIR) that controls the PUSH and POP operation input port connections.

Table 18-16 lists the asynchronous FIFO user interface signals for PolarPro II devices. A short description of each signal and its port characteristics are provided.

Table 18-17: PolarPro II FIFO Signal Descriptions

Signal	Signal Type	Description
Clk1	Input	Clock input.
P1	Input	PUSH or POP signal depending on DIR. Default: PUSH
ALMOST_FULL	Output	Almost full indicator. NOTE: This flag is asserted when FIFO has room for only 1 (ALMOST_FULL) or FULL.

© 2008 QuickLogic Corporation

Table 18-17: PolarPro II FIFO Signal Descriptions (Continued)

<table>
<tr><th>Signal</th><th>Signal Type</th><th>Description</th></tr>
<tr><td>PUSH_FLAG</td><td>Output</td><td>FIFO push level indicator. The following table lists its values and descriptions.
<table>
<tr><td>0000</td><td>Full</td></tr>
<tr><td>0001</td><td>Empty</td></tr>
<tr><td>0010</td><td>Room for more than ½</td></tr>
<tr><td>0011</td><td>Room for more than ¼</td></tr>
<tr><td>0100</td><td>Room for less than ¼ to/or 64</td></tr>
<tr><td>1010</td><td>Room for 32 to 63</td></tr>
<tr><td>1011</td><td>Room for 16 to 31</td></tr>
<tr><td>1100</td><td>Room for 8 to 15</td></tr>
<tr><td>1101</td><td>Room for 4 to 7</td></tr>
<tr><td>1110</td><td>Room for 2 to 3</td></tr>
<tr><td>1111</td><td>Room for 1</td></tr>
<tr><td>Others</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>Clk2</td><td>Input</td><td>Clock input.</td></tr>
<tr><td>P2</td><td>Input</td><td>PUSH or POP signal depending on DIR.

Default: PUSH</td></tr>
<tr><td>ALMOST_EMPTY</td><td>Output</td><td>Almost empty indicator. This signal is asserted when the FIFO has one left or is EMPTY.</td></tr>
<tr><td>POP_FLAG</td><td>Output</td><td>FIFO pop level indicator. The following table lists its values and descriptions
<table>
<tr><td>0000</td><td>Empty</td></tr>
<tr><td>0001</td><td>1 entry in FIFO</td></tr>
<tr><td>0010</td><td>At least 2 entries in FIFO</td></tr>
<tr><td>0011</td><td>At least 4 entries in FIFO</td></tr>
<tr><td>0100</td><td>At least 8 entries in FIFO</td></tr>
<tr><td>0101</td><td>At least 16 entries in FIFO</td></tr>
<tr><td>0110</td><td>At least 32 entries in FIFO</td></tr>
<tr><td>1000</td><td>Less than ¼ to 64 full</td></tr>
<tr><td>1101</td><td>¼ or more full</td></tr>
<tr><td>1110</td><td>½ or more full</td></tr>
<tr><td>1111</td><td>Full</td></tr>
<tr><td>Others</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>FIFO_EN</td><td>Input</td><td>FIFO enable. If asserted, block functions as a FIFO. If not asserted, block functions as a RAM.</td></tr>
</table>

NOTE: ROM module generation is not supported for PolarPro, PolarPro II and ArcticLink devices.

© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation

Chapter 19
PCI Configuration for 32-bit Parts

This chapter describes PCI register configuration and contains the following sections:

- **“Functional Overview” on page 255**
- **“Configuring PCI Registers” on page 256**

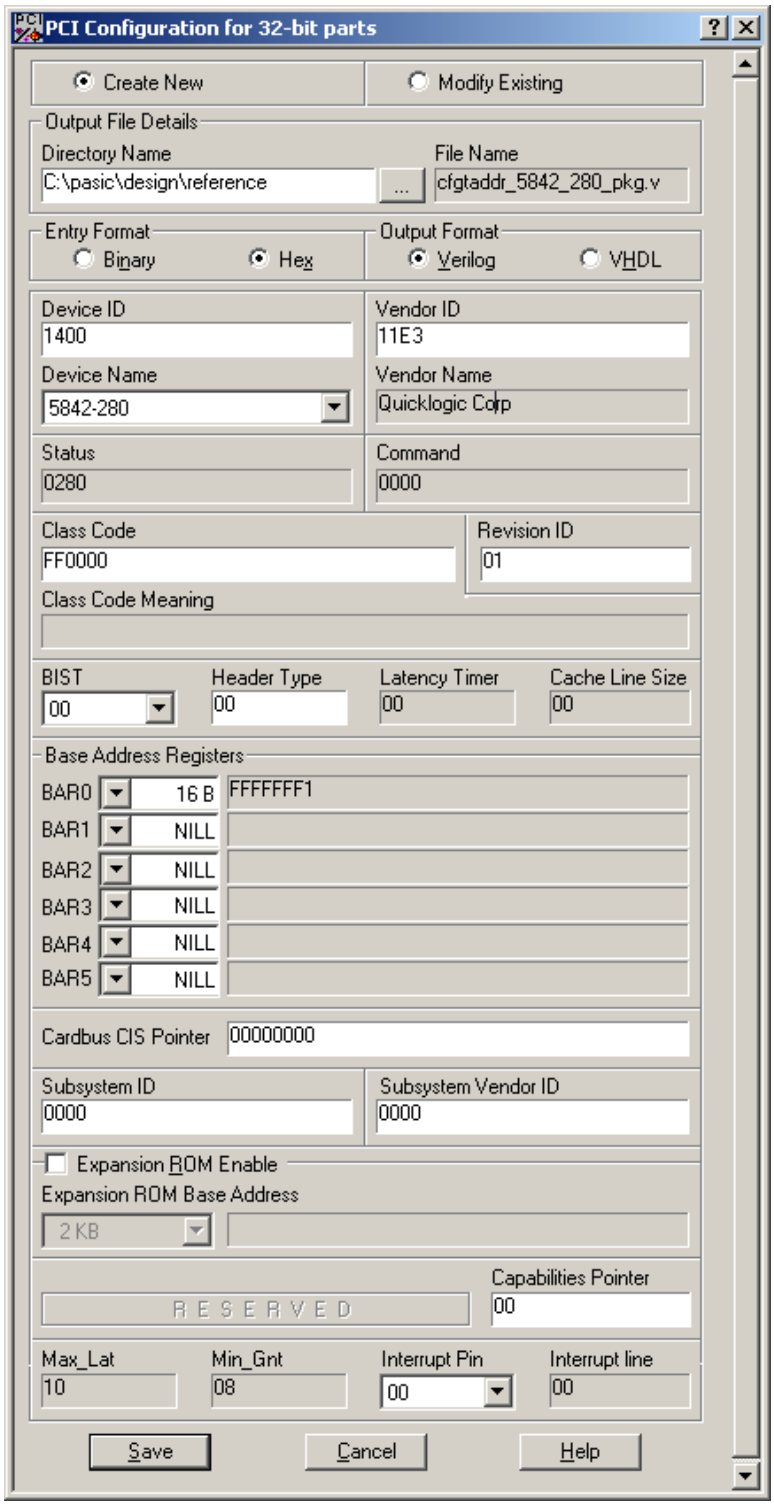
19.1 Functional Overview

The *PCI Configuration for 32-bit parts* interface is used in automatic updating of the configuration parameters of the 32-bit PCI parts in SpDE. According to the PCI standard specifications, there are configuration parameters that need to be set. The designers can use the default parameters or can change the values. The default parameters are stored in HDL package files (`cftgaddr_<part_name>_pkg.vhd` for VHDL based designs and in `cftgaddr_<part_name>_pkg.v` for Verilog designs). These package files in turn are used by the PCI designs. This interface helps in creating new package files or modifying the existing package files.

© 2008 QuickLogic Corporation

19.2 Configuring PCI Registers

Table 19-1 gives a brief description of each register to be configured in the PCI Configuration for 32-bit parts dialog box. For complete details of each register refer to *PCI Specifications, v.2.2*. The first three items are GUI-specific. This dialog box can be accessed by selecting **Tools>PCI Configuration** from the SpDE menu bar.



© 2008 QuickLogic Corporation

Table 19-1: PCI Configuration for 32-bit Parts Options

Option	Function
Input/Output File Details	Directory: Directory in which the input file is read and output file is generated. File Name: The name of the Verilog/VHDL package file to be read or generated.
Entry Format	The data entry format (binary or hex).
Output Format	The format of the file to be read or generated (Verilog or VHDL).
Device ID	This field identifies the particular device. The vendor allocates this identifier. ID is validated against the list of valid quick logic IDs. If the ID is not found in the list, a warning is displayed and the option is given whether to accept or reject the entered ID.
Device Name	If it is a QuickLogic device the name of the device is displayed after the ID is entered. Otherwise it will be blank. The user may also select the device name from the list and the device ID is displayed automatically.
Vendor ID	This field identifies the manufacturer of the device. Valid vendor identifiers are allocated by the PCI SIG to ensure uniqueness.
Vendor Name	Displays the vendor name of the vendor ID entered in Vendor ID box.
Status	The Status register is used to record status information for PCI bus related events. The layout of the register is shown in **Figure 19-1**. This is displayed for information. The user cannot modify it.
Command	The Command register provides coarse control over a device's ability to generate and respond to PCI cycles. **Figure 19-2** shows the layout of the register. This is displayed for information. The user cannot modify it.
Class Code	The Class Code register is used to identify the generic function of the device. The Class Code is validated against the available list of codes and the meaning is displayed if it is available. If the code is not found in the list, a warning is displayed and the option is given whether to accept or reject the entered code.
Revision ID	This register specifies a device specific revision identifier. There is no special validation for this field.
BIST	This optional register is used for control and status of Built-in Self Test (BIST). The valid values are listed; the user must select one.
Header Type	This byte identifies the layout of the second part of the predefined header (beginning at byte 10h in Configuration Space) and also whether the device contains multiple functions. There is no special validation for this field.
BAR0-5	The user can select size of the 6 Base Address Registers. The user does not need to enter the decoding part. Once the size is selected the code is displayed automatically.
CardBus CIS Pointer	For devices to share silicon between CardBus and PCI use this optional register. The field is used to point to the Card Information Structure (CIS) for the CardBus card. There is no special validation for this field.
Subsystem ID	These registers are used to uniquely identify the expansion board or the subsystem where the PCI device resides. They provide a mechanism for the expansion board vendors to distinguish their boards from one another even though the boards may have the same PCI controller.

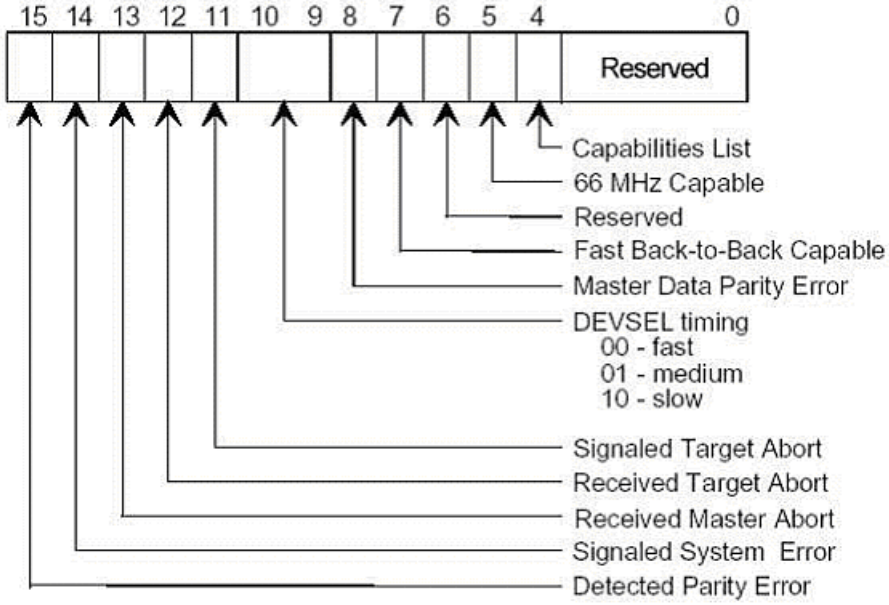
© 2008 QuickLogic Corporation

Table 19-1: PCI Configuration for 32-bit Parts Options (Continued)

Option	Function
Subsystem Vendor ID	These registers are used to uniquely identify the expansion board or the subsystem where the PCI device resides. They provide a mechanism for the expansion board vendors to distinguish their boards from one another even though the boards may have the same PCI controller.
Expansion ROM Enable and Base Address	Some PCI devices, especially those intended for use on expansion boards on PC architectures, require local EPROMs for expansion ROM. A four-byte register (see **Figure 19-3**) at offset 30h in a type 00h predefined header is defined to handle the base address and size information for this expansion ROM. Check the Expansion ROM Enable checkbox to enable the ROM base address. Select the size from the Expansion ROM Base Address pull-down list. The corresponding code is displayed automatically.
Max_Lat and Min_Gnt	Read-only byte registers used to specify the device settings for Latency Timer values. For both registers, the value specifies a period of time in units of one-fourth microsecond. These two fields are display only. The default values are displayed.
Interrupt Pin	The Interrupt Pin register selects which interrupt pin the device (or device function) uses. A value of 1 corresponds to INTA#. A value of 2 corresponds to INTB#. A value of 3 corresponds to INTC#. A value of 4 corresponds to INTD#. Devices (or device functions) that do not use an interrupt pin must put a 0 in this register. The user must select a value from the list.
Interrupt Line	This 8-bit register defines to which system interrupt request line the interrupt output is routed

Figure 19-1 describes the Status Register.

Figure 19-1: Status Register



© 2008 QuickLogic Corporation

Figure 19-2 describes the Command Register.

Figure 19-2: Command Register

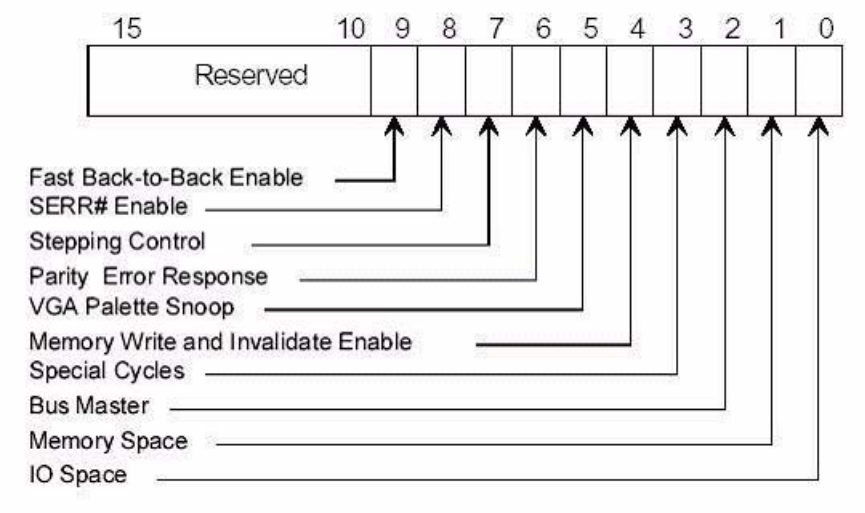
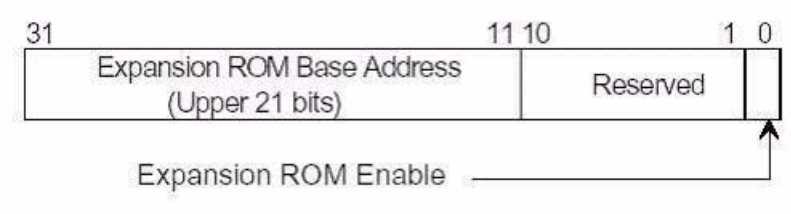


Figure 19-3 describes the Expansion ROM Enable and Base Address Register.

Figure 19-3: Expansion ROM Enable and Base Address Register



© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation

Chapter 20
PLL/CCM Wizard

This chapter describes the PLL/CCM Wizard tool and contains the following sections:

- **"Functional Overview" on page 261**
- **"Running PLL/CCM Wizard" on page 262**
- **"Device Family Selection" on page 262**
- **"Setting Up PLLs" on page 263**
- **"Output Settings" on page 266**

The PLL/CCM Wizard is a configuration tool for the Phase Locked Loop (PLL) and Configurable Clock Managers (CCMs). It can generate Verilog and VHDL in a user-specified directory.

20.1 Functional Overview

All QuickLogic's 0.25 um devices have embedded PLLs to enable users to reduce the clock-to-out timing of their designs and to run the internal logic of the device at a faster or slower rate than the incoming clock frequency.

PLLs compensate for a delay in the clock network. As a result, there is no clock delay between the input clock signal and the clock signal coming from the clock network or driving off-chip devices. Using time domain multiplexing, PLLs enable users to improve device area efficiency by sharing resources within the device.

QuickLogic devices have two different PLL types:

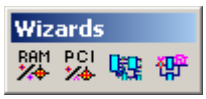
- **PLL**—user-programmable PLLs that can be programmed for clock frequency multiplication and division, and can be used to improve design I/O timing.
- **CCM for PolarPro device family**—configurable Clock Managers (CCMs) enable flexible control and management of clock frequency, phase shift and skew.

These PLLs can be used in eight distinct modes of operation (see **Table 5-9 on page 73**).

© 2008 QuickLogic Corporation

20.2 Running PLL/CCM Wizard

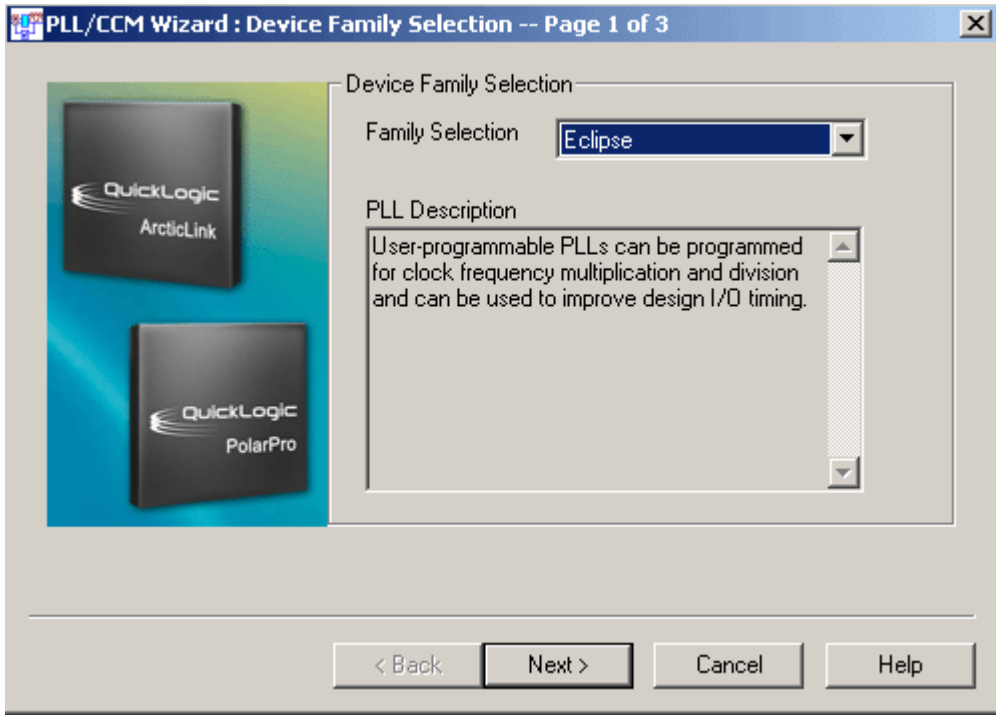
The PLL/CCM Wizard can be accessed either from **Tools > PLL/CCM Wizard** or from the toolbar.



1. Choose a target family.
2. Depending on the family selected, the corresponding PLL is configured. Various configurable parameters are displayed and should be set accordingly.
3. Select the Verilog or VHDL (or both) as output.
4. Select the output directory. Files to be generated are displayed along with warnings (in cases when files already exist).
5. Once the configuration is completed, dump the output in the specified directory.

20.3 Device Family Selection

To start using the PLL/CCM Wizard, choose a device family to define the PLL/CCM type. Currently, there is one PLL and one type of CCM supported by PolarPro and other device families.

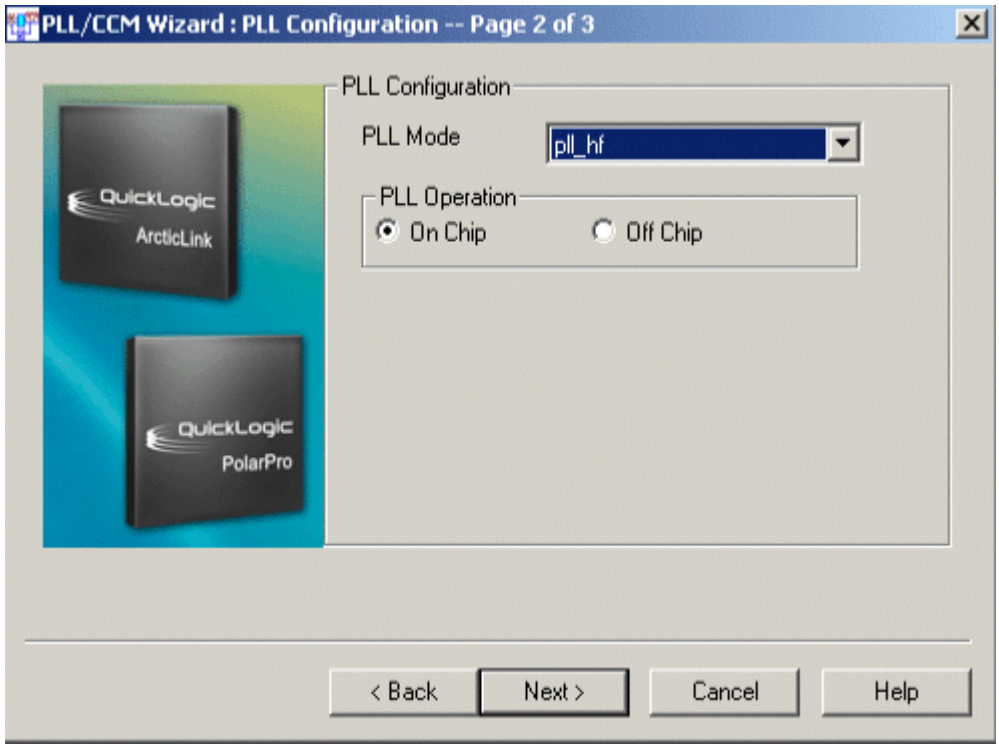


- **Family Selection**—list of device families to determine the PLL/CCM type.
- **PLL/CCM Description**—description of the PLL/CCM used in the selected device family.

© 2008 QuickLogic Corporation

20.4 Setting Up PLLs

Each of the PLL/CCM types has different sets of configuration parameters.



- **PLL Mode**—PLLs can be used in eight modes of operation (*see* **Table 5-9 on page 73**).
- **On Chip**—clock will be used inside the chip.
- **Off Chip**—clock will be used outside the chip.

20.4.1 PLL Configuration

Table 20-1 lists the PLL configuration parameters.

Table 20-1: PLL Modes

Mode Name	PLLIN	PLLOUT	I/P Freq Range (MHz)
pll_hf	X	X	66 – 150
pll_lf	X	X	25 – 66
pll_mult2hf	X	2X	33 – 125
pll_mult2lf	X	2X	12.5 – 33
pll_div2hf	X	.5X	250 – 500
pll_div2lf	X	.5X	50 – 250
pll_mult4	X	4X	12.5 – 50
pll_div4	X	.25X	100 – 500

© 2008 QuickLogic Corporation

NOTE: When PLL is configured for *on-chip* mode, it becomes the user's responsibility to connect the output signals properly in the design.

NOTE: For a detailed description on PLL modes, refer to **"Phase Lock Loop (PLL) Macros" on page 73**.

20.4.2 CCM Configuration for the PolarPro Device Family

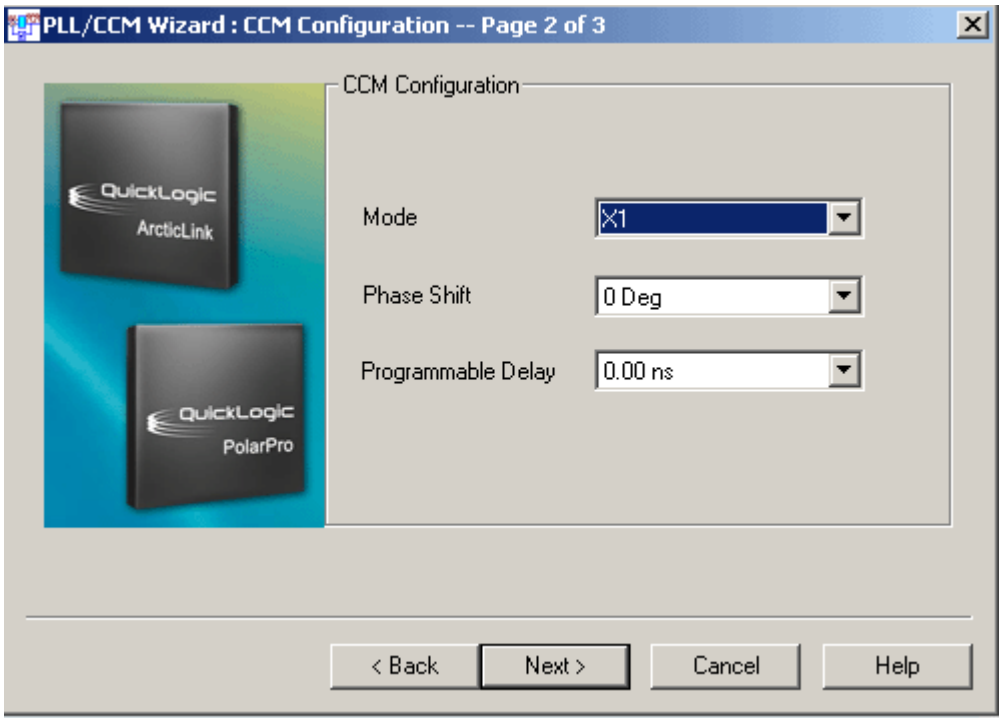


Table 20-2 lists the configuration parameters for the PolarPro device family CCM.

Table 20-2: CCM Modes for the PolarPro Device Family

Mode	Description
X1	Multiply by 1 frequency mode
X2	Multiply by 2 frequency mode
X4	Multiply by 4 frequency mode

Table 20-3 lists the phase shift values for the PolarProdevice family CCM.

Table 20-3: CCM Phase Shift for the PolarPro Device Family

Phase Shift (Degrees)	Description
0	Phase shift of 0 degrees in Clock Output
90	Phase shift of 90 degrees in Clock Output
180	Phase shift of 180 degrees in Clock Output
270	Phase shift of 270 degrees in Clock Output

© 2008 QuickLogic Corporation

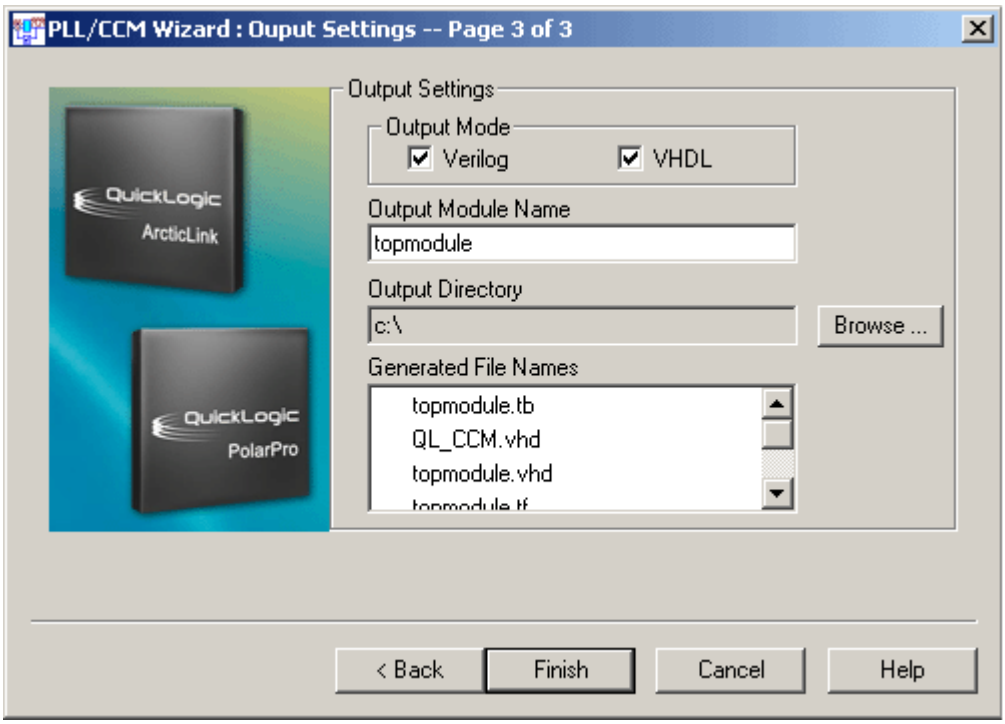
Table 20-4 lists the phase shift values for the PolarPro device family CCM.

Table 20-4: CCM Programmable Delay for the PolarPro Device Family

Programmable Delay (ns)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
2.50

© 2008 QuickLogic Corporation

20.5 Output Settings



Define all output settings that will create the required PLL configuration files:

- **Output Mode**—generates the output files in Verilog or VHDL (or both).
- **Output Module Name**—module name used for PLL configuration.
- **Output Directory**—output directory where all the files will be created.
- **Generated File Names**—list of files that will be created.

NOTE: The PLL/CCM Wizard support is the same for PolarPro, PolarPro II and ArcticLink devices.

© 2008 QuickLogic Corporation

Chapter 21
Constraint Manager

This chapter describes the features of the Constraint Manager. It contains the following sections:

- **“Functional Overview” on page 267**
- **“Using Constraint Manager” on page 268**

21.1 Functional Overview

The Constraint Manager is a unified user interface to the design constraint editors (i.e., Fix Placement, Configuration and Timing Constraints editors). Refer to the chapter **“Design Constraints and Analysis” on page 287** for details on usage of the editors. The Constraint Manager works with the Window-Based Placement user interface (now supported in V340HPC). This feature gives designers the flexibility to enter the design constraints before or after the design has been placed and routed.

The Constraint Manager provides following features:

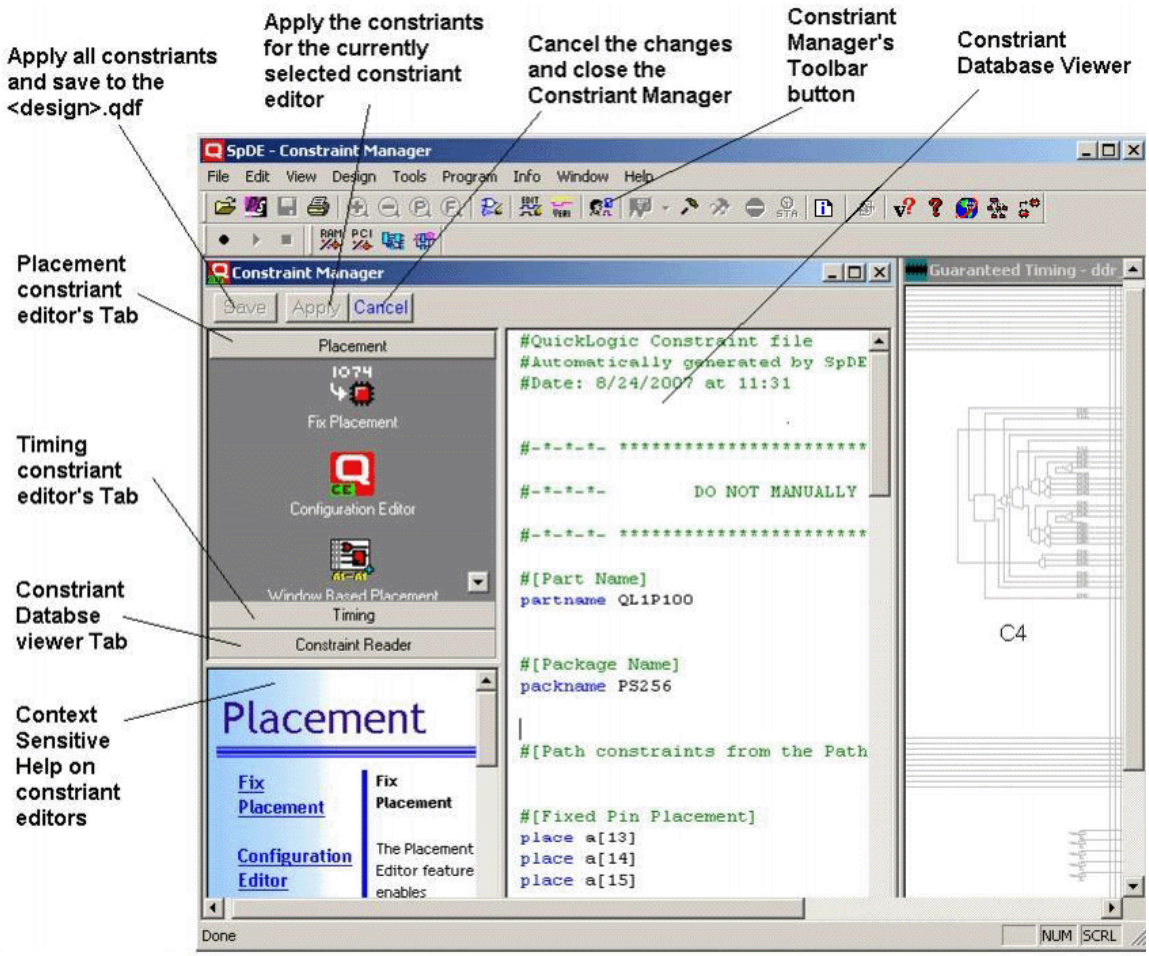
- A single point interface to add constraints to the user design.
- Single level interface to QuickLogic Constraint File (`.qcf`).
- Constraint Database Viewer, with color scheme for text display.

NOTE: The Constraint Manager and Static Timing Analyzer tools cannot be used simultaneously. However, this feature will be available in a future version of QuickWorks.

© 2008 QuickLogic Corporation

21.2 Using Constraint Manager

The Constraint Manager provides the designer with the ability to easily apply constraints and check the placed and routed results, without closing the constraint editor(s) or writing back to the QuickLogic constraint file (`.qcf`). The Constraint Manager has a constraint database viewer to view a snapshot of the constraint file at any intermediate level. The designer can choose to save the constraint file by clicking **Save**. The Constraint Manager can be accessed using the SpDE toolbar button or by selecting **Tools>Constraint Manager** from the SpDE menu bar.



The Constraint Manager has the following three tabs:

- Placement
- Timing
- Constraint Reader

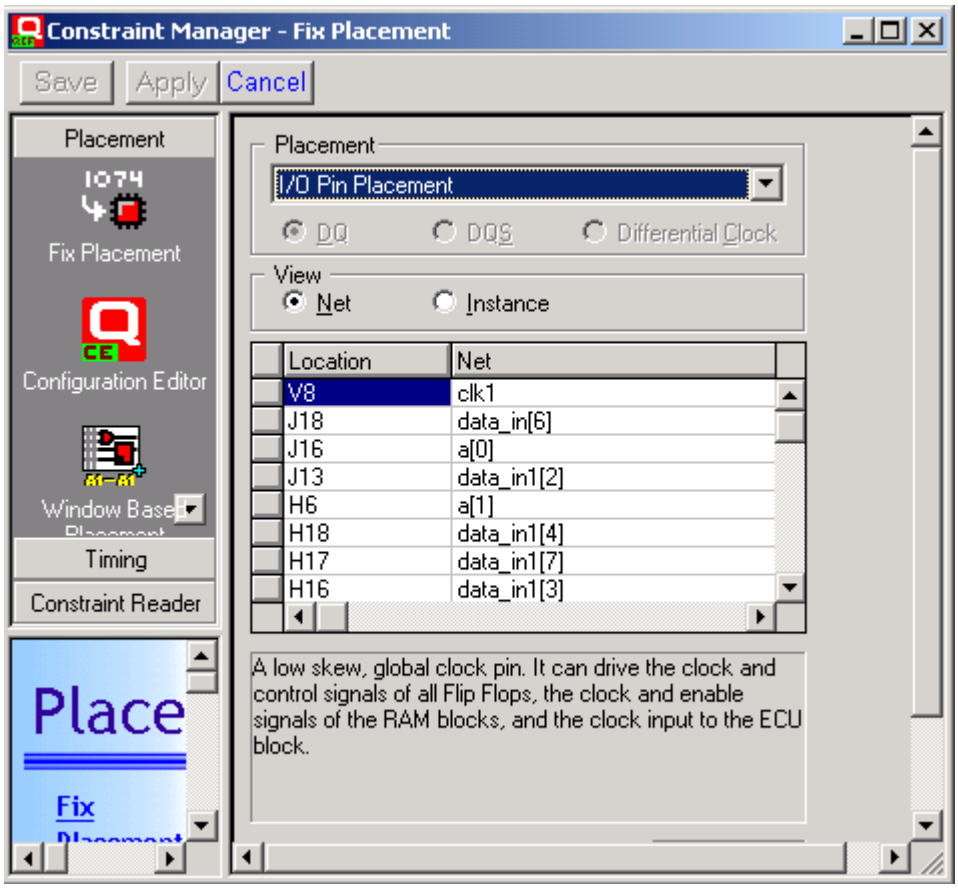
© 2008 QuickLogic Corporation

21.2.1 Placement Tab

The Placement tab contains all the supported placement constraints editor tools such as Fix Placement Editor, Configuration Editor and Window-Based Placement Editor.

21.2.1.1 Fix Placement Editor

For more details on the Placement Constraint editor(s), refer to **“Placement Constraints” on page 296**.



21.2.1.2 Configuration Editor

NOTE: The Configuration Editor is not available for pASIC devices.

The Configuration Editor allows users to configure I/O architectural features of QuickLogic devices.

21.2.1.2.1 I/O Configuration for Pinnacle, Pre-Pinnacle and Eclipse Device Families

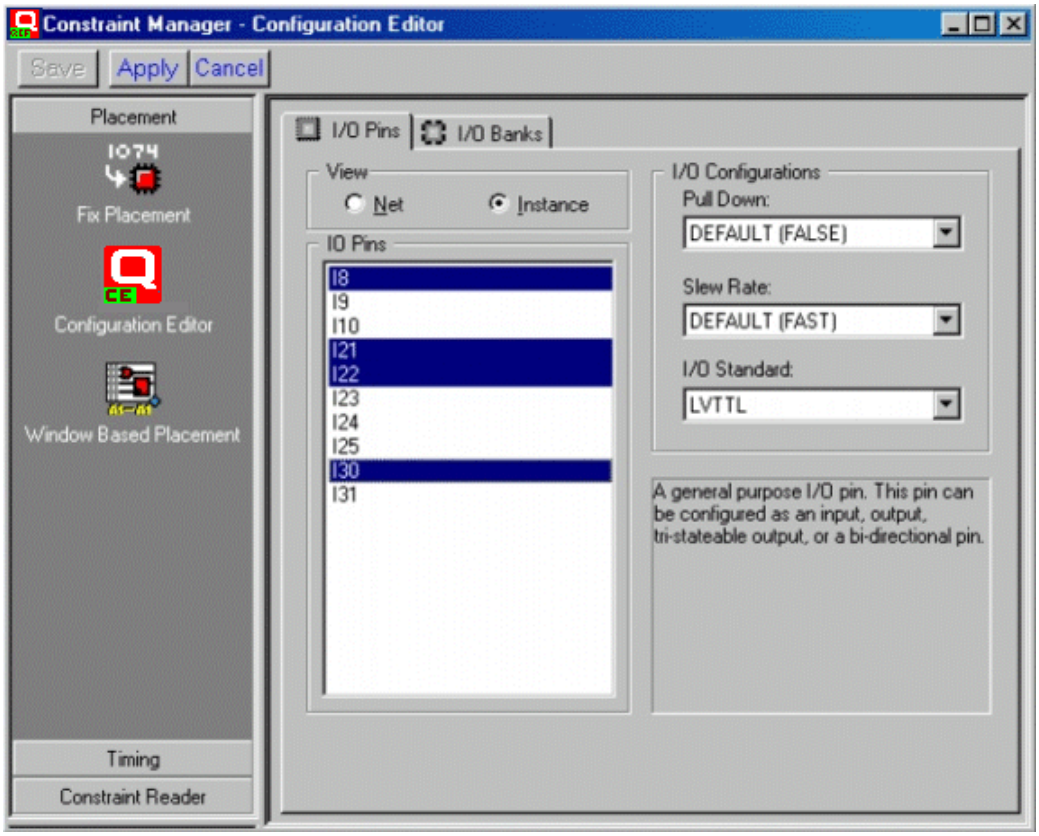
The Configuration Editor has two tabbed sections:

- I/O Pins
- I/O Banks

I/O Pins: The user interface provides a list of I/O pins and possible I/O configurations. The I/O pins can be listed as instance names or net names, by selecting **Instance** or **Net** in the View field. The designer can select the desired I/O pin(s) for configuration and specify the required I/O configuration settings, such as pull down (yes/no), slew rate (fast/normal), and I/O Bank standard. Multiple pins can be selected for a configuration setting. The designer can also view a brief description of the selected I/O pin.

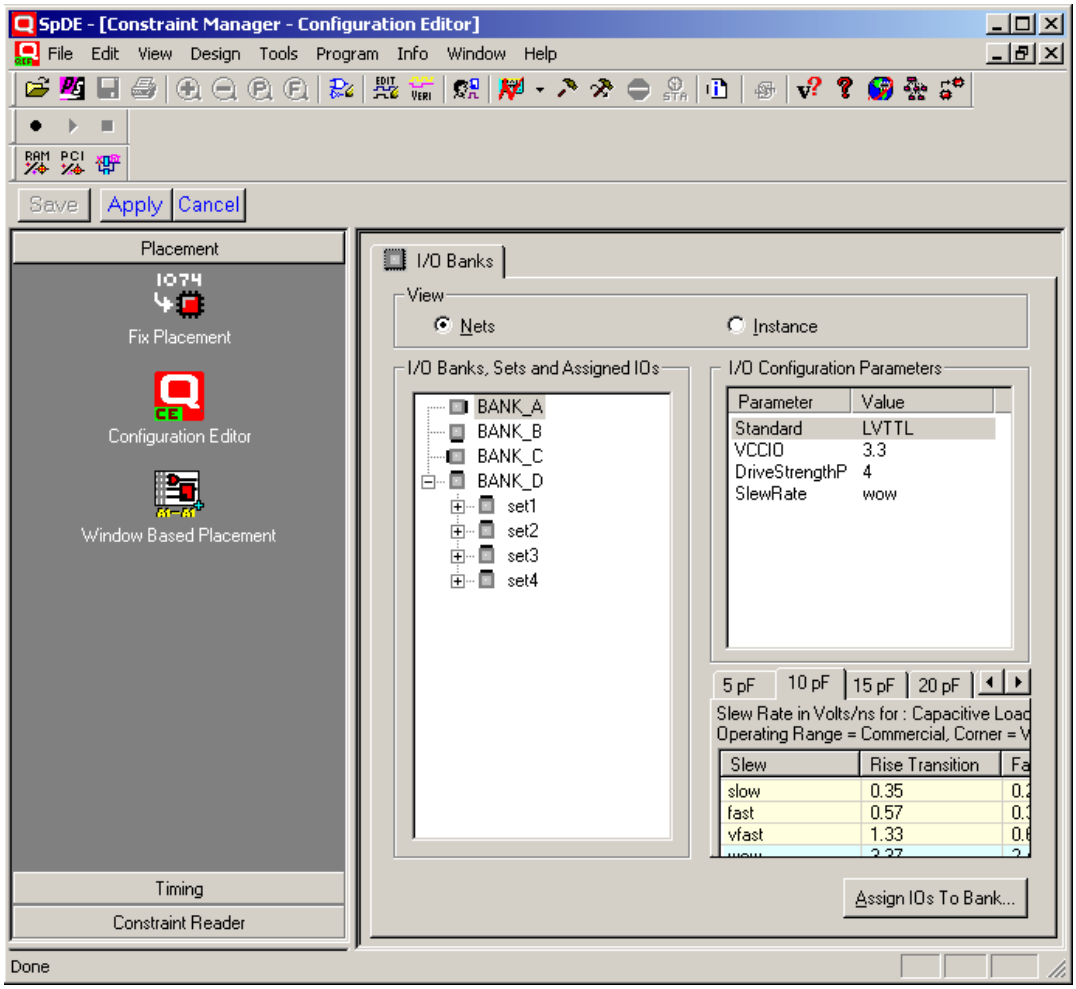
Each I/O also supports slew rate control. The default slew rate is fast. However, the user can specify on an individual I/O whether an I/O has fast or slow slew rate.

NOTE: The fast slew rate is the default for SpDE.



© 2008 QuickLogic Corporation

I/O Banks: The Configuration Editor provides a separate option within the same user interface to allow editing of I/O standards on a bank-by-bank basis. A graphical representation of the device shows all the supported banks of the targeted device for a design. The designer can select the bank(s) and choose the I/O standard from a drop-down list of I/O Bank Standards. Multiple banks can be selected for a configuration setting. The designer can also view a brief description of the selected I/O bank.



I/Os can be configured in the following ways:

- LVTTL (default standard)
- PCI
- LVCMOS2
- GTL+
- SSTL3
- SSTL2

© 2008 QuickLogic Corporation

Table 21-1 lists various legal combinations of these standards within an I/O bank.

Table 21-1: I/O Pin Configurations

Bank	Pins								
	LVTTL	PCI	LVCMOS2	GTL+		SSTL2		SSTL3	
				I/P	O/P	I/P	O/P	I/P	O/P
GTL+			P	P	P				
SSTL2			P		P	P	P		
SSTL3	P	P	P		P			P	P

21.2.1.2.2 I/O Configuration for the PolarPro Device Family

Configuration for the PolarPro Device:

Unlike other devices that have to configure pins and banks, for the PolarPro device, the banks section is enhanced to configure both banks and pins together. In configuring the PolarPro device I/Os, the following terminology and hierarchy is used:

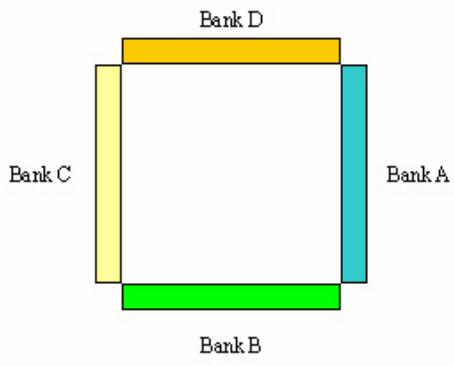
- Banks
- Sets
- I/Os
- Data (DQ)
- Clock
- Strobe (DQS)

DDRIOs are grouped as banks and sets in the following way:

- Banks consist of sets
- Sets consist of DDRIOs
- DDRIOs consists of data (DQ), clock, and strobe (DQS)

I/Os are divided into various banks. The number of banks is device dependent. Each bank of I/O has an independent VCCIO, and can therefore support different I/O standards within a device.

Figure 21-1: VCCIO Bank Configuration



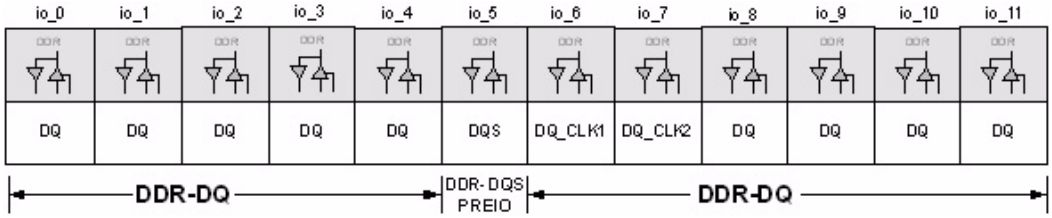
© 2008 QuickLogic Corporation

There are two types of I/Os: DDR I/O and General Purpose I/O (GPIO). DDR I/Os are placed in the Bank D domain. Bank D is divided into sets of 12 DDRs as follows:

- 9x DDR-DQ configured for DQ and DM signals
- 1x DDR-DQS
- 2x DDR-DQ configurable as DQ or differential input/output clocks

DQS strobe signal is used in reading in and writing out data. Every set of DDR I/Os has its own DQS signal to be used with 9 to 11 DDR-DQ I/Os in the group. The signals are arranged as shown in **Figure 21-2**.

Figure 21-2: DDR-IO Set



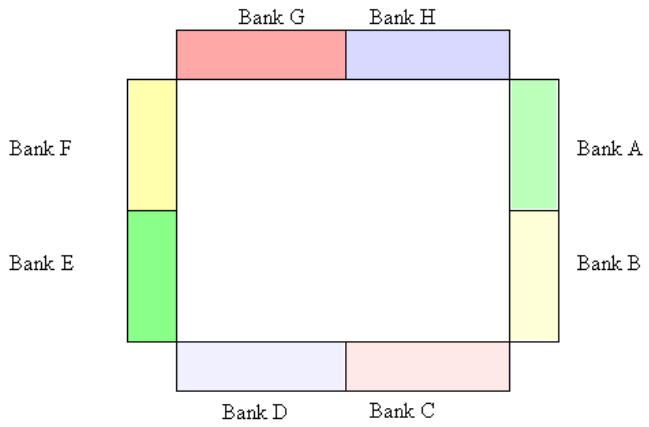
Configuration for the PolarPro II Device:

I/Os and I/O Banks of the PolarPro II device family are configurable. The following terminologies are used in configuration of I/O banks and I/Os:

- Banks
- I/Os

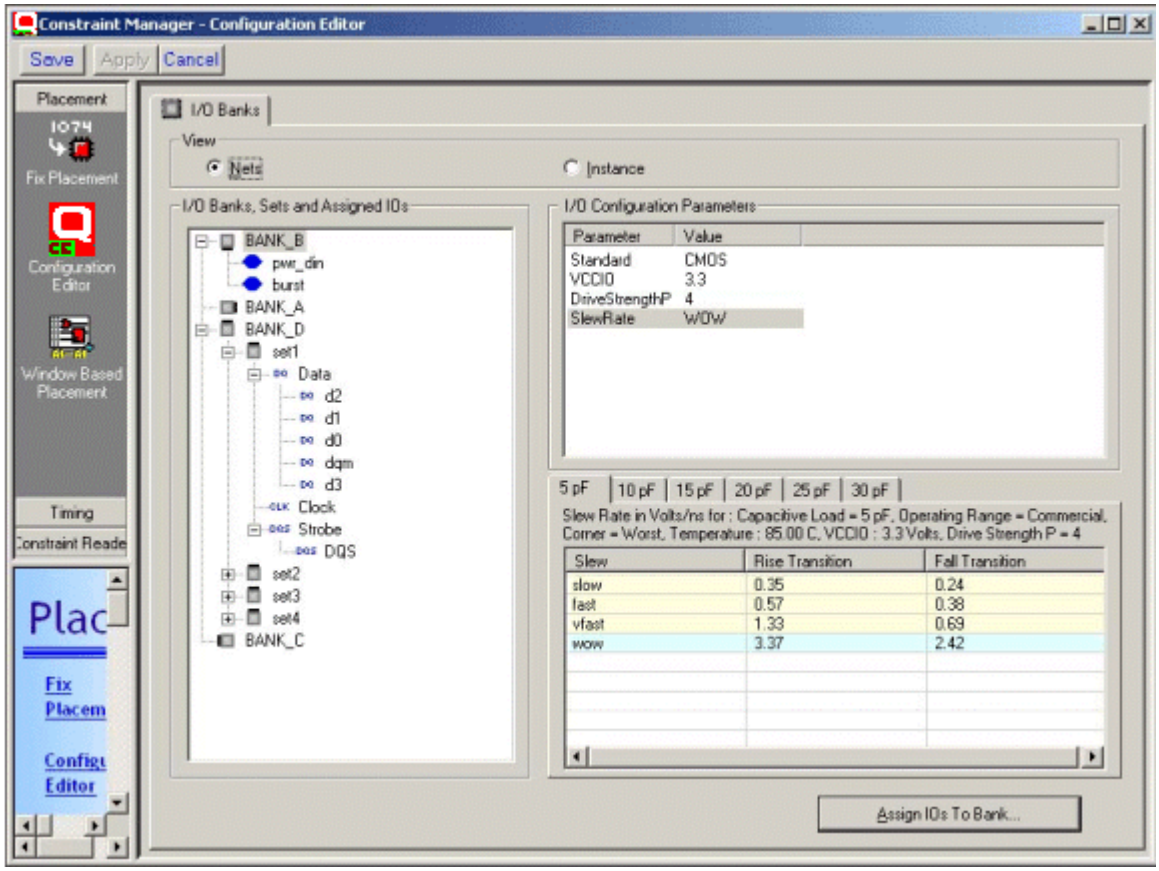
I/Os are divided into various banks. The number of banks is device dependent. Each bank of an I/O has an independent VCCIO, and can therefore support different I/O standards within a device.

Figure 21-3: VCCIO Bank Configuration for PolarPro II Devices



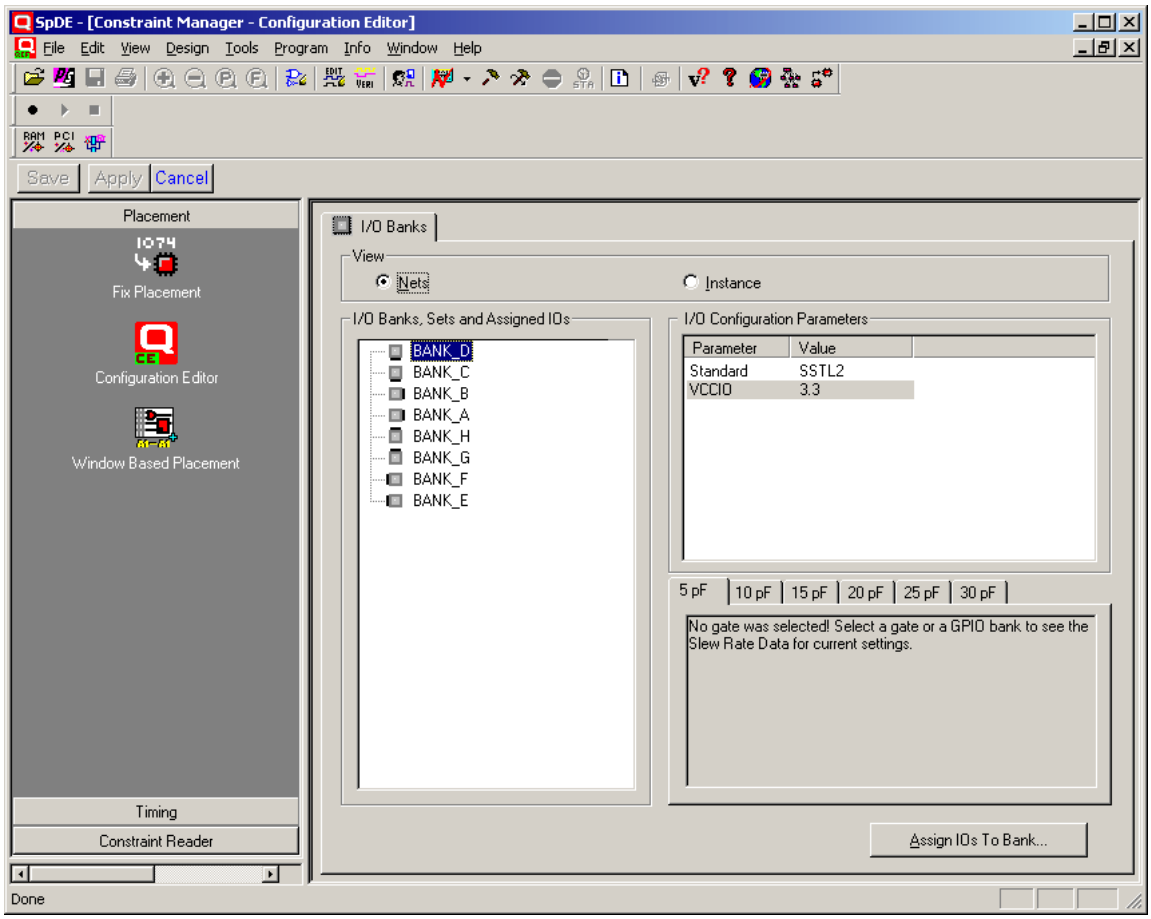
21.2.1.2.3 Configuring the PolarPro and PolarPro II Devices I/Os using GUI

A design targeted to a PolarPro device will have an I/O configuration page as shown below. This page displays a hierarchy for banks, sets, data, clock, strobe, and I/Os.



© 2008 QuickLogic Corporation

A design targeted to a PolarPro II device will have an I/O configuration page as shown below. This page displays a hierarchy for banks and I/Os.



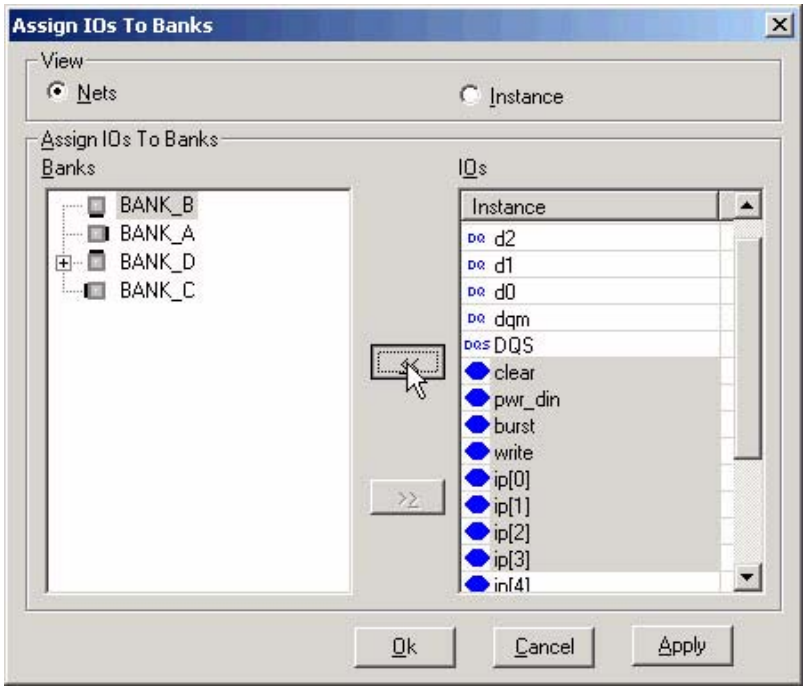
To assign I/Os to GPIO banks or DDRIO sets, click **Assign IOs To Bank** and then supply I/O configuration parameters, shown on the right-hand side as a list with applicable values. Various parameters are applicable at different levels in the hierarchy. Designers can choose a value from the combo list. To view the list of I/O pins in the form of instance or net names, select either **Nets** or **Instance** option in the View field. On the bottom right-hand side, the output values for Slew Rate (in V/ns) are displayed. The Slew Rate (V/ns) shown here varies with Capacitive Load, Operating Range, Corner settings, VCCIO and Drive strength P. Operating Range and Corner settings can be made using the **Delay Modeler** settings in the **Tools >Options** menu.

NOTE: To change the default configuration settings of the I/O gates, it is required to pull I/Os in banks and sets.

© 2008 QuickLogic Corporation

To configure I/Os in Bank_A, Bank_B or Bank_C of the PolarPro device; or to configure I/Os in Bank_A, Bank_B or Bank_C, Bank_D, Bank_E or Bank_F, Bank_G, or Bank_H of the PolarPro II device:

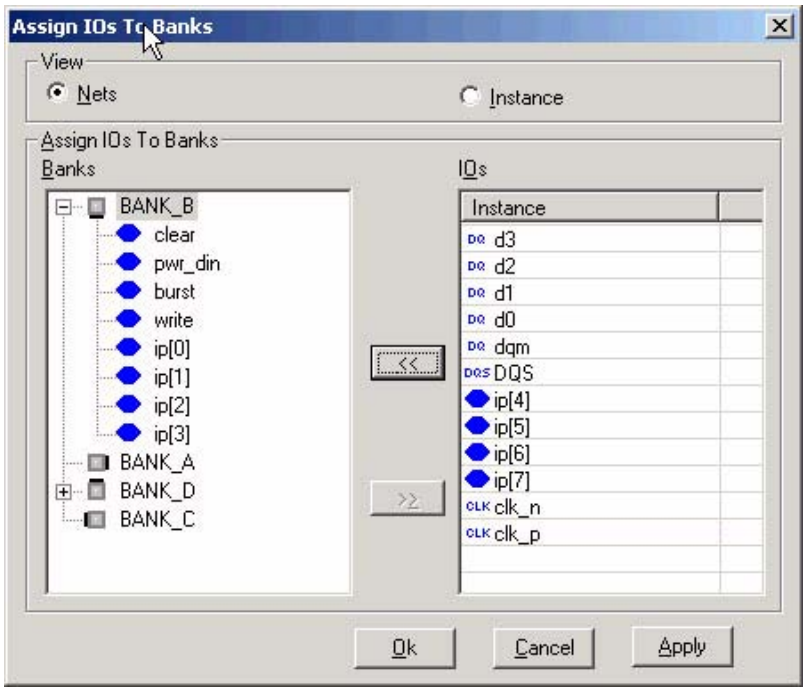
1. Click **Assign IOs To Bank**. The Assign IOs To Bank dialog box opens.



2. Select a bank from the Banks list.
3. Select an I/O from the I/Os list.

© 2008 QuickLogic Corporation

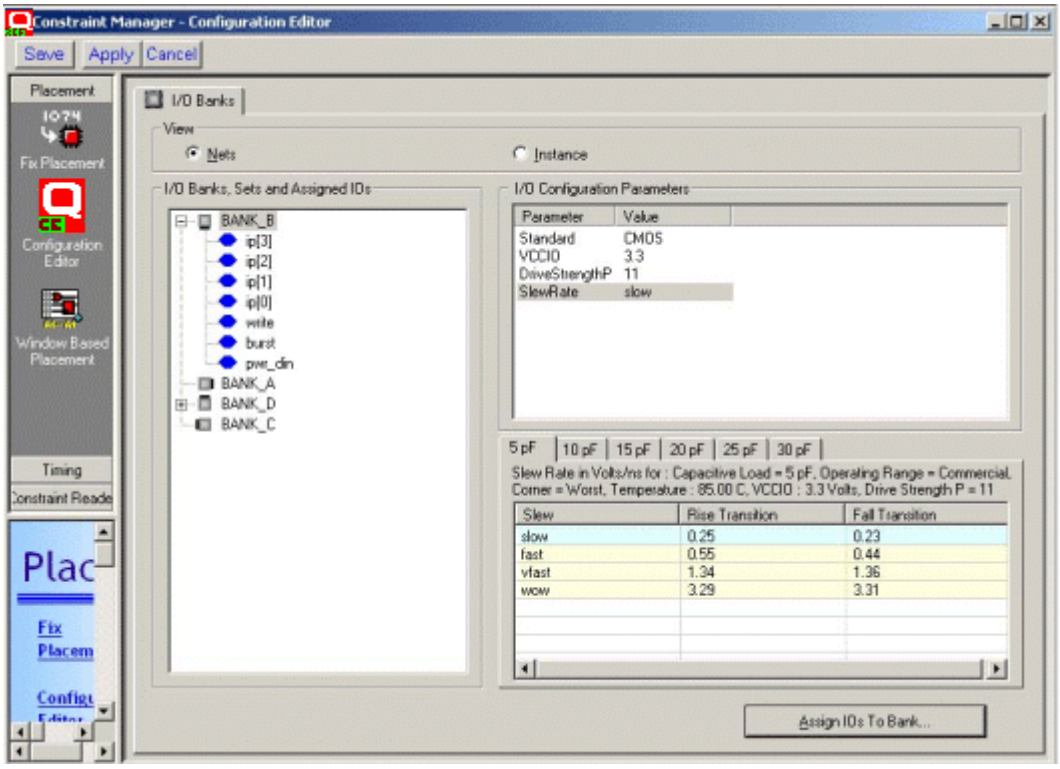
4. Click the << button to assign the selected I/Os to the selected bank.



5. To remove already assigned I/Os, select them and click the >> button.
6. Select a bank and configure the VCCIO, I/O Standard, Drive Strength P, and Slew Rate parameters. Choose values from the combo list.
7. Select I/Os and configure the PLL Down and PLL Up parameters.
8. Click **Apply** to apply I/O configuration constraints to the design and continue the Assign operation.
9. Click **OK** to apply the assignments and close the Assign IOs To Bank dialog box.

© 2008 QuickLogic Corporation

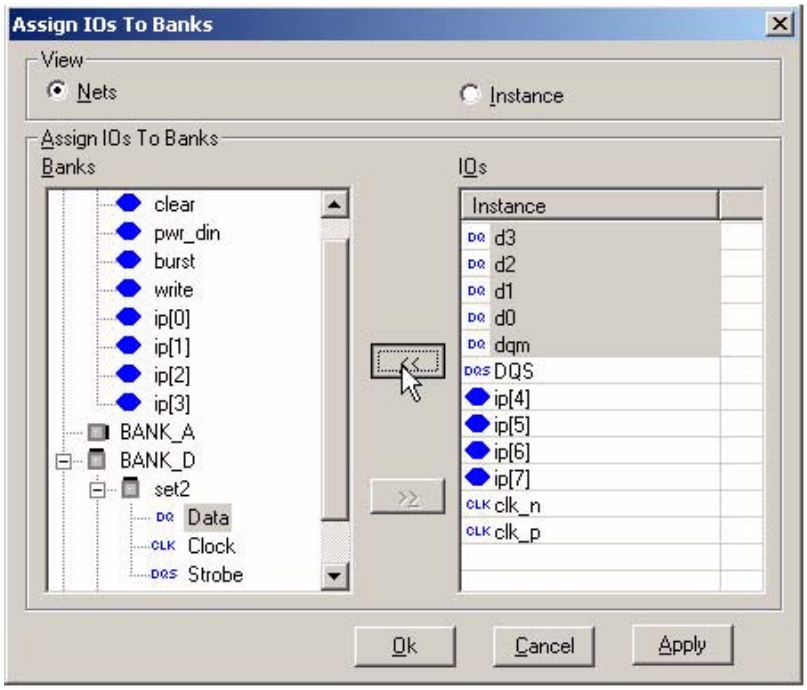
10. Click **Save** to save the constraints.



© 2008 QuickLogic Corporation

To configure I/Os for sets:

1. Click **Assign IOs To Bank**. The Assign IOs To Bank dialog box opens.



2. Select the **Data** set from Bank_D to be configured.
3. Select the DQ I/Os (DQ) from the IOs list.
4. Click the << button to assign the selected I/Os to the selected set.
5. Repeat steps 3 and 4 for Strobe (DQS) and Clock (CLK) sets.

   NOTE: Only bounded sets are shown for Bank D; other sets are not listed.

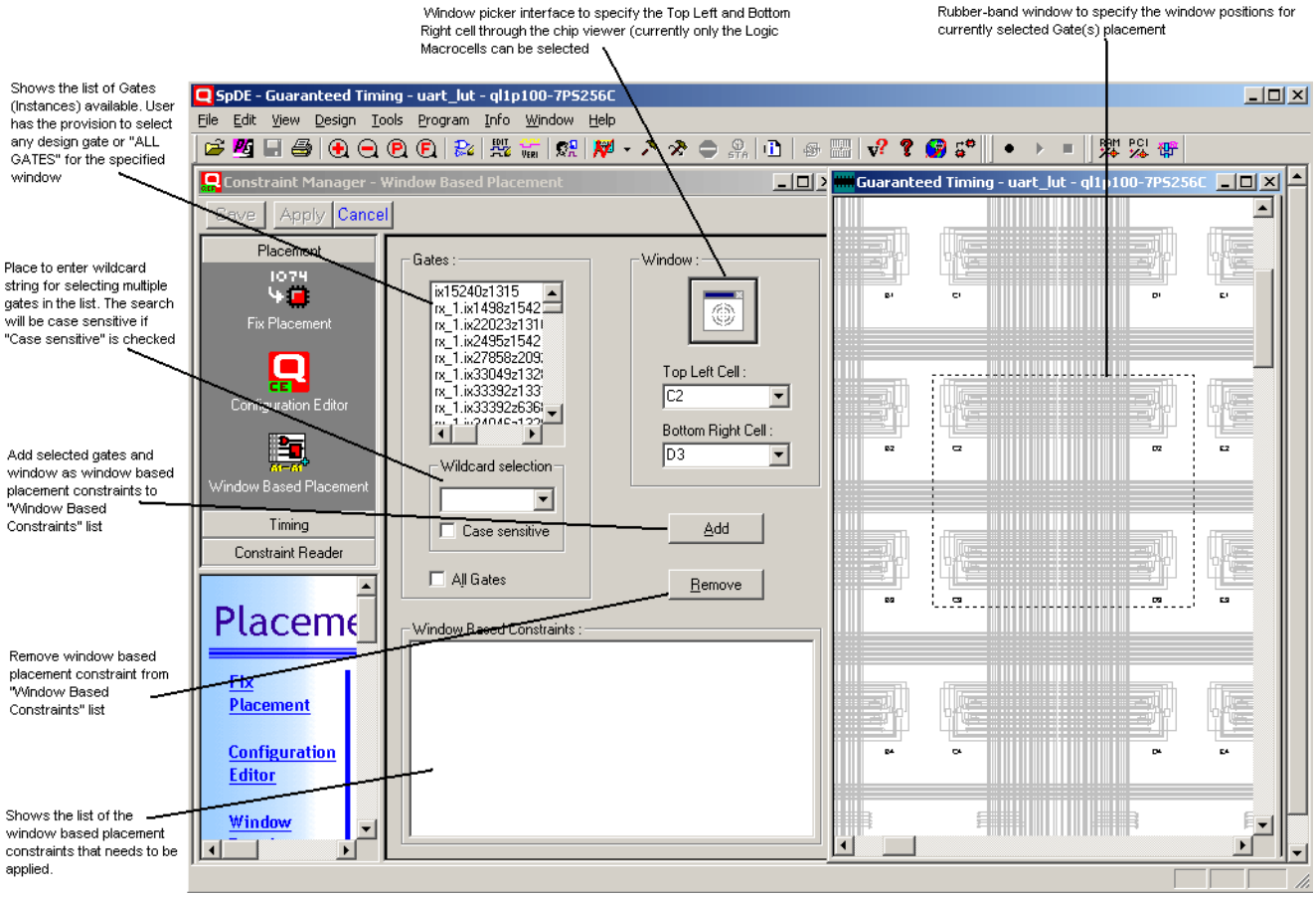
6. Configure the VCCIO and IO Standard parameters for a chosen set. Select values from the combo list.
7. Select the Data/Strobe/Clk group and configure the Drive Strength P, Drive Strength N and Slew Rate parameters.
8. Select the I/Os and configure the Pull Up parameter.
9. Click **Apply** to apply I/O configuration constraints to design.
10. Click **OK** to apply the assignments and close the Assign IOs To Bank dialog box.
11. Click **Save** to save the constraints.

    NOTE: Once a DQ I/O, a strobe or a clock has been assigned to a set, a GPIO cannot be assigned. To reset the database after removing all DQs, strobes and clocks, click **Apply** before assigning a GPIO. Similarly, when all GPIOs have been removed from a set, click **Apply** to reset the database.
    On clicking **Apply** or **OK**, the verifier automatically assigns all other required DQ and DQS I/Os. Users are not required to assign all DQ I/Os or strobes.

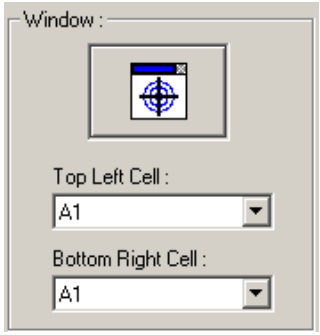
© 2008 QuickLogic Corporation

21.2.1.3 Window-Based Placement Editor

The Window-Based Placement Editor user interface provides a mechanism to specify and list the placement constraints (i.e., specifying a rectangular window for one or more design gates [instances]). Click **Add** or **Remove** to update the list of window-based placement constraints applied for the design.


Window picker interface to specify the Top Left and Bottom Right cell through the chip viewer (currently only the Logic Macrocells can be selected
Rubber-band window to specify the window positions for currently selected Gate(s) placement
Shows the list of Gates (Instances) available. User has the provision to select any design gate or "ALL GATES" for the specified window
Place to enter wildcard string for selecting multiple gates in the list. The search will be case sensitive if "Case sensitive" is checked
Add selected gates and window as window based placement constraints to "Window Based Constraints" list
Remove window based placement constraint from "Window Based Constraints" list
Shows the list of the window based placement constraints that needs to be applied.
SpDE - Guaranteed Timing - uart_lut - ql1p100-7PS256C
File Edit View Design Tools Program Info Window Help
Constraint Manager - Window Based Placement
Guaranteed Timing - uart_lut - ql1p100-7PS256C
Save Apply Cancel
Placement
Fix Placement
Configuration Editor
Window Based Placement
Timing
Constraint Reader
Gates :
Wildcard selection
Case sensitive
All Gates
Window :
Top Left Cell :
C2
Bottom Right Cell :
D3
Add
Remove
Window Based Constraints :
Fix Placement
Configuration Editor
Window

The designer can specify a window (i.e., a rectangular section of macrocells in the chip viewer, defined by top-left and bottom-right corners) as placement constraints to the selected Gate or All Gates. Addition of window-based placement constraints is also provided through a *rubber band* window (window selection).



© 2008 QuickLogic Corporation

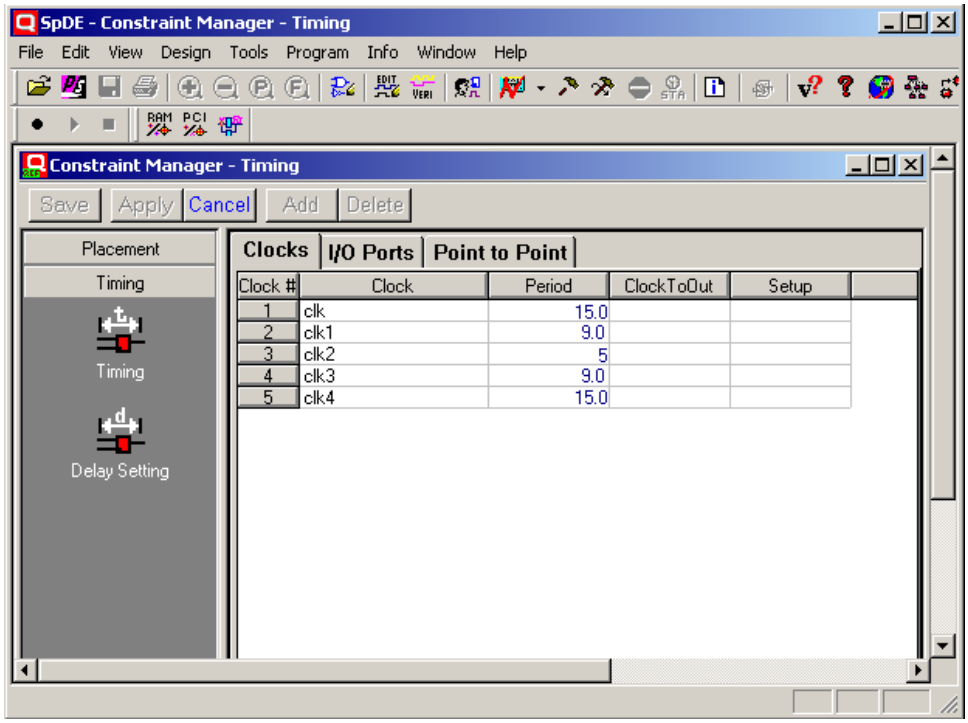
21.2.2 Timing Tab

The Timing tab contains the Timing Constraint Editor tool. For more details on the tool, refer to **“Timing Constraints” on page 288**.

21.2.2.1 Timing Constraint Editor

21.2.2.1.1 Clocks Tab

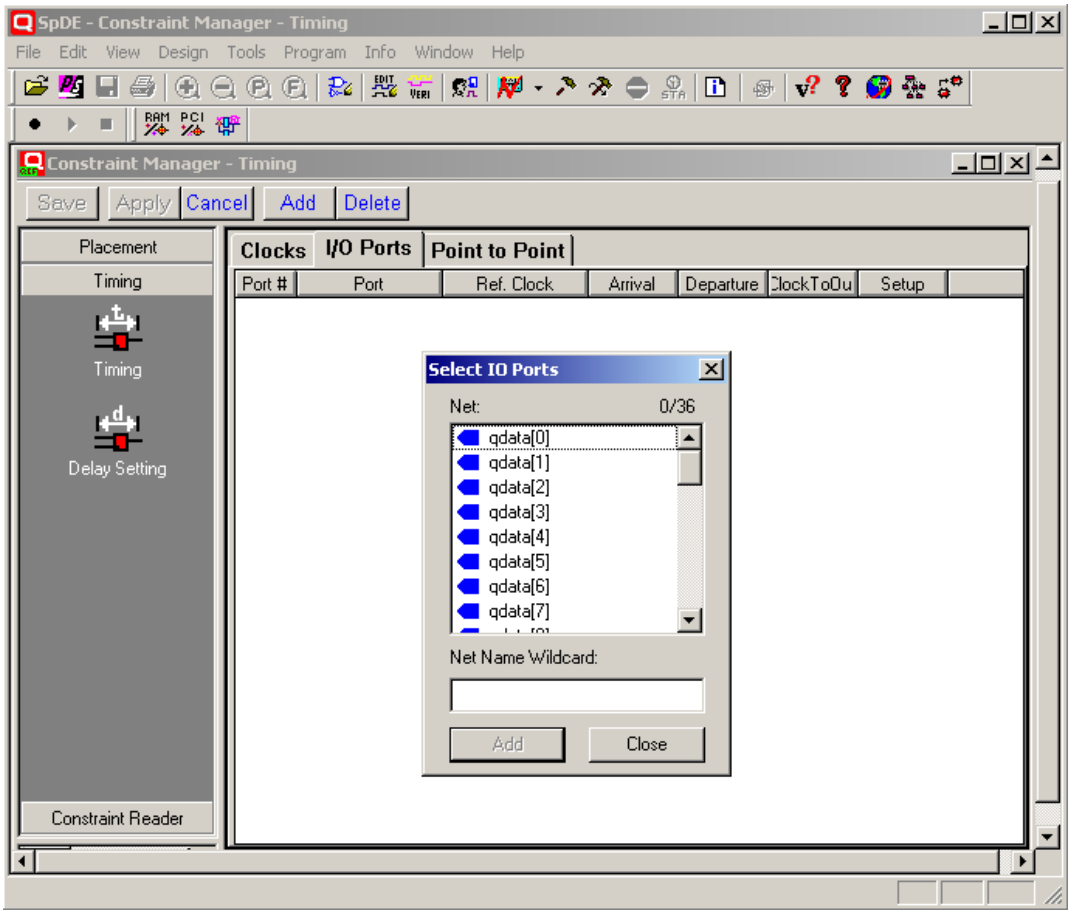
Clocks is the first tab in the Timing Constraint Editor and it is used to specify timing constraints for the clock signal(s) (both direct and user-defined clocks) driving flip-flops in the design (i.e., Period, Clock-to-Out, Setup time). The user cannot add user-defined clocks in the Constraint Manager. This can be done only through the `.qcf` file.



21.2.2.1.2 I/O Ports Tab

Ports is the second tab in the Timing Constraint Editor and it is used to specify timing constraints to general I/O signal(s) in a design (i.e., Arrival and Setup time for input signals, Departure and Clock-to-Out for output signals).

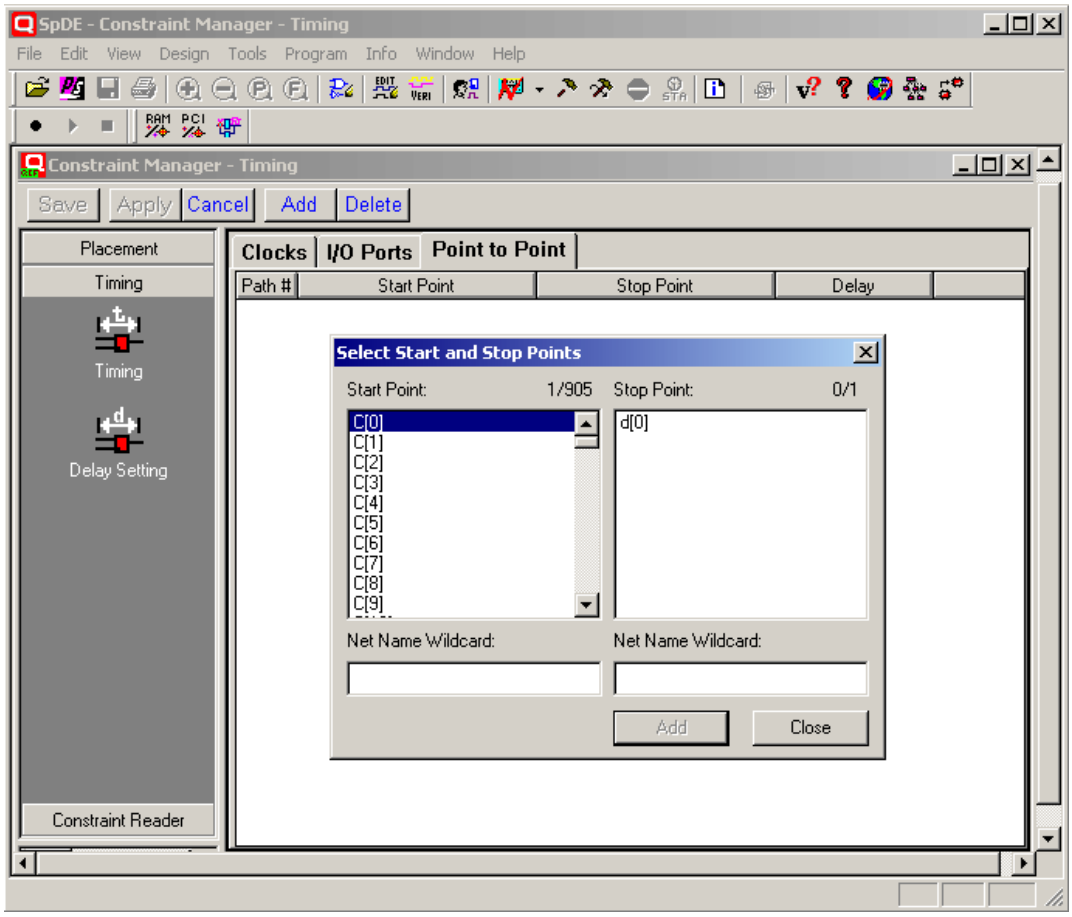
NOTE: The timing constraints for the selected I/O signals in the design are specified with respect to a reference clock signals.



© 2008 QuickLogic Corporation

21.2.2.1.3 Point-to-Point Tab

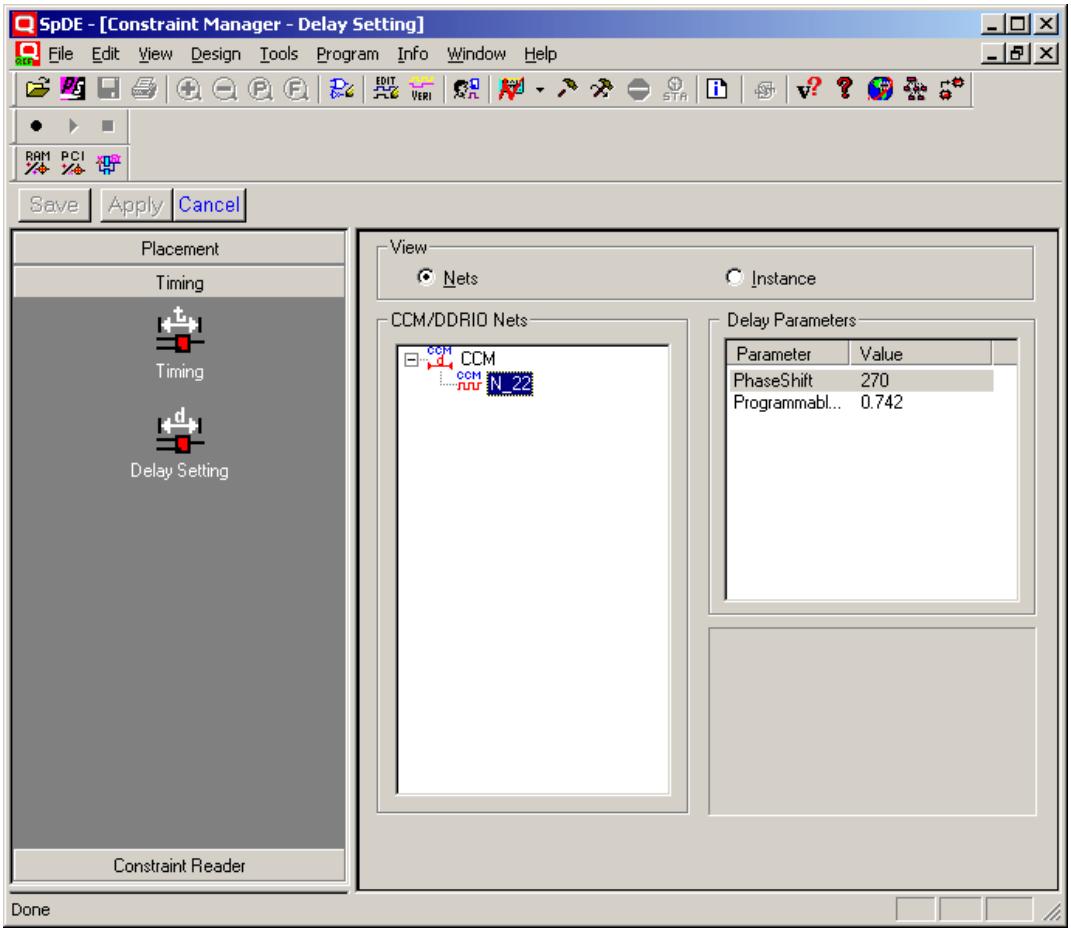
Point-to-Point is the third and last tab in the Timing Constraints Editor and it is used to specify timing constraints for any paths having a start and stop point in a design (i.e., Point-to-Point delay).



21.2.2.2 Delay Setting

The Delay Setting interface enables setting the phase shift and delay values in the CCM and DDRIO blocks.

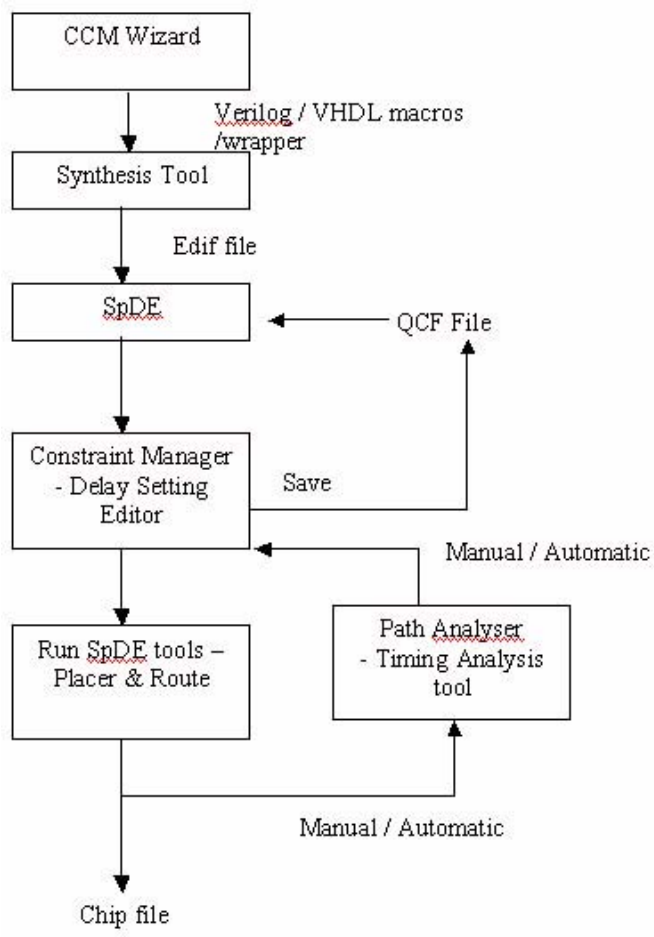
Net or Instance names are used as handles to these settings. All the delay parameters are displayed in the right-hand side frame. For example, as shown below, when a net name under CCM is selected, the Programmable Delay and Phase Shift are displayed as parameters.



For any synthesized design, the initial values for these delay settings will be coming from the `.edif` file. Before synthesis, users can set CCM delay settings using the CCM Wizard. Users have the option of setting these values using the `.qcf` file at any time. Using the interface above, users can set these values on the fly before and after running the Place & Route. Users can use the **Apply** and **Save** buttons to apply and retain these values. Once saved, these setting are retained in `.qcf` files, and if the design is saved, the setting values are written into the chip file as well.

© 2008 QuickLogic Corporation

Figure 21-4: Delay Settings Flow



21.2.3 Constraint Reader Tab

The Constraint Reader tab contains the Constraint Database Viewer tool. The Constraint Database Viewer is a snapshot of the constraint file.

NOTE: The Constraint Database Viewer is not an editor. The editing feature will be available in a future version of QuickWorks.

21.2.4 Constraint Manager Toolbar

The Constraint Manager has a toolbar with five buttons: **Save**, **Apply**, **Cancel**, **Add** and **Delete**. (**Add** and **Delete** buttons are displayed only for the Timing Constraint Editor.)

When the designer adds or modifies one or more constraints in any of the design constraint editors, the **Apply** and **Save** buttons are enabled.

© 2008 QuickLogic Corporation

When the designer applies any design constraints and clicks **Apply** from the Constraint Manager toolbar, the constraint database is updated for the currently selected constraint editor and the database viewer shows the applied constraints in color-coded, sectioned text format.

When the designer clicks **Save** from the Constraint Manager toolbar, the constraint database gets updated for all the Design Constraint editors and writes back the text from the constraint database viewer to the `<design>.qcf`. (The **Save** command is an *apply all constraints and save them to a file* command.)

The **Cancel** command closes and exits the Constraint Manager.

NOTE: By default, the Constraint Manager opens up showing the Constraint Database viewer.

NOTE: The **Apply** button is linked with individual design editors and applies currently selected constraint editor's constraints to the constraint database, while the **Save** button applies the constraints in all constraint editors and writes back the constraints to `<design>.qcf`.

© 2008 QuickLogic Corporation

Chapter 22
Design Constraints and Analysis

This chapter discusses the following aspects of design constraints and analysis:

- **“Design Constraints” on page 287**
- **“Drag-and-Drop Functionality” on page 304**
- **“Fix Placement Using the QCF File” on page 306**
- **“Fixing Unused Pins” on page 308**
- **“Highlight Net and Static Timing Analyzer” on page 309**
- **“Highlight a Net” on page 310**
- **“Clock Network Usage” on page 311**
- **“Report File” on page 312**

22.1 Design Constraints

The Design Constraint Editor is fully integrated into SpDE, providing designers with the flexibility to enter design constraints before or after the design has been placed and routed. The editor is easy to use and it also eliminates the need to manipulate text files several steps back in the design flow.

This section demonstrates how to apply Timing and Placement constraints to your design.

For more information, see also:

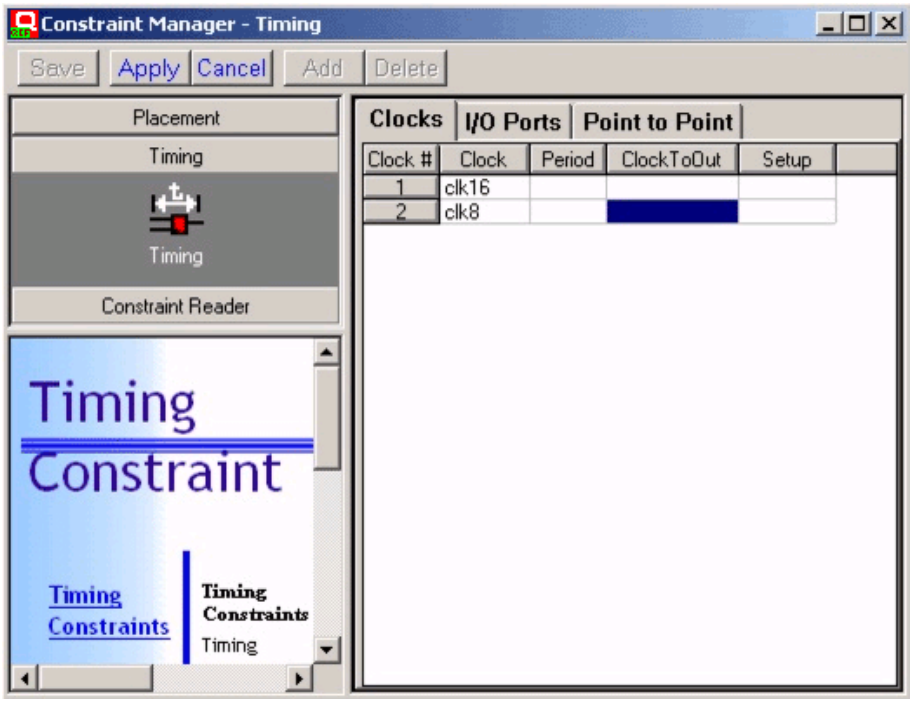
- **“Using Constraint Manager” on page 268**

© 2008 QuickLogic Corporation

22.1.1 Timing Constraints

To launch the Timing Constraints Editor:

1. Select **Tools>Constraint Manager>Timing>Timing** in the Design Editor.



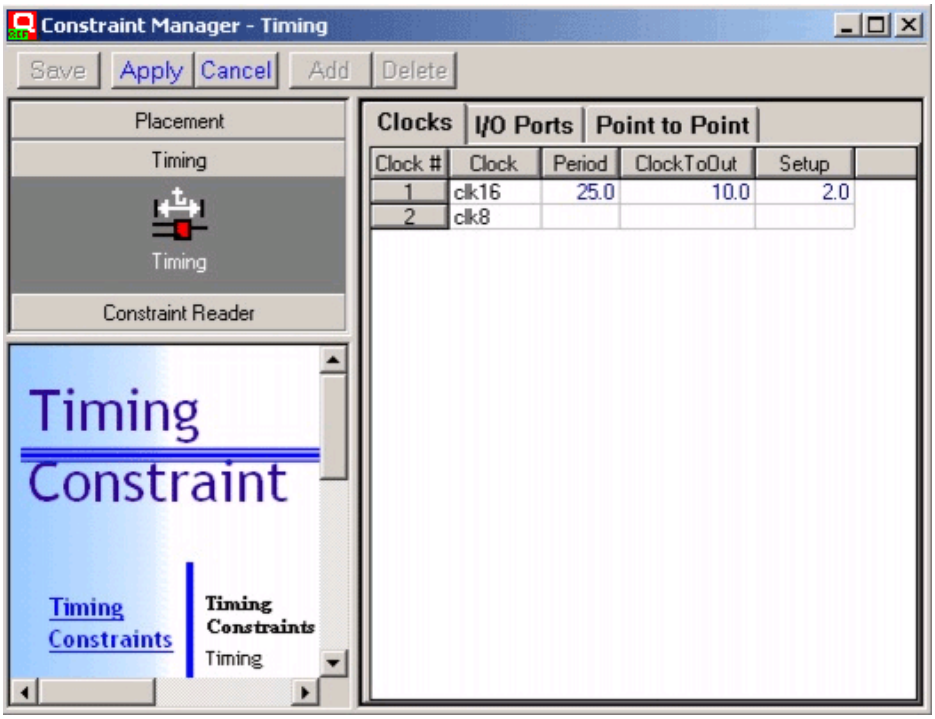
For more information, see also:

- **"Using Constraint Manager" on page 268**

© 2008 QuickLogic Corporation

22.1.1.1 Clock Signals

The first tab in the Timing Constraints Editor is used to specify timing constraints for the clock signal(s) as well as any user-defined clocks in the design. Click in the **Period** cell that corresponds to the appropriate clock signal. Enter the period of the clock signal (in nanoseconds), and press **Enter**. For example, if your clock signal `CLK16` is toggling at a rate of 40 MHz, type `25` in the Period box.



If any of your clock signals have flip-flops with critical Clock-to-Out timing, you may wish to enter those constraints in the editor. This is done the same way you entered the Period value. Click in the **Clock-to-Out** column for the appropriate clock signal, enter a value and then press **Enter**. A value of `10 ns` is specified in the example.

The final constraint to apply to the clock signal is the setup time. Entering this constraint will assist the placer tool in finding the optimal placement of the flip-flops. Click in the **Setup** column, enter the value and then press **Enter**. A value of `2 ns` is specified in the example.

22.1.1.2 I/O Signals

To apply similar constraints to general I/O signals:

1. In the Timing Constraints dialog box, select the I/O Ports tab and click **Add**. This opens the Select I/O Ports dialog box.
2. Select any of the I/O signals for your design.

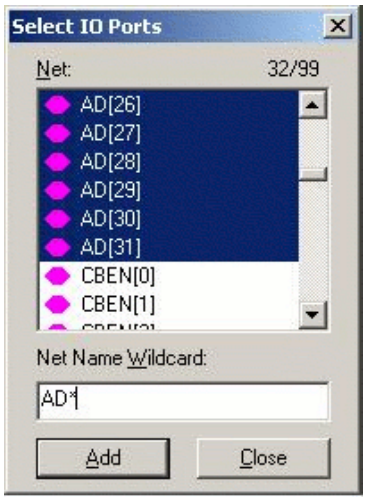
   NOTE: Designers have the flexibility to select only the signals that have critical timing constraints.

3. To add a signal to the Timing Constraint Editor, select the signal and click **Add**.

© 2008 QuickLogic Corporation

22.1.1.2.1 Net Name Wildcard

Another helpful option is the Net Name Wildcard dialog box. By entering a string of text and proper usage of the `*/?` character, you can add several I/O ports simultaneously.
The figure below shows how to add all of the signals that begin with the letters `A16`. When you enter the text and * character, the interface automatically highlights all of the ports or flip-flops that meet the wildcard criteria.



22.1.1.2.2 Deleting Signals

To delete any signals from the list, click the button next to the selected signal in the Path # column, or Port # column. The name varies depending on which mode you are in (i.e., I/O Ports or Point-to-Point). In the Timing Constraints editor select the path and click **Delete** to remove the signal.

22.1.1.3 Timing Constraints for I/O Ports

This section defines the different timing constraints that can be applied to I/O ports – **Figure 22-1 on page 291** shows these timing constraints.

22.1.1.3.1 Arrival

The Arrival time is the time it takes for a signal to go from the output of an external device into the input of the QuickLogic device. For example, the time for a signal to arrive at the QuickLogic device from an external flip-flop is the arrival time.

22.1.1.3.2 Setup

The Setup time is the time it takes for a signal to travel from the pads of a device to the input of the an internal flip-flop, subtracting the time it takes for the clock signal to travel from a clock pad to the clock input of a flip-flop. Refer to **"Path Analysis" on page 322** to see how setup time is calculated with SpDE.

© 2008 QuickLogic Corporation

22.1.1.3.3 Clock-to-Out

The Clock-to-Out time is the sum of the time it takes a clock signal to travel from the clock pad to the flip-flop, and the time it takes the output of the flip-flop to reach the output pad of the device.

22.1.1.3.4 Departure Time

The departure time is the amount of time it takes for a signal to travel from the QuickLogic device to an external device such as a flip-flop.

Figure 22-1: Diagram of Arrival and Departure Times (Setup and Clock-to-Out are internal)

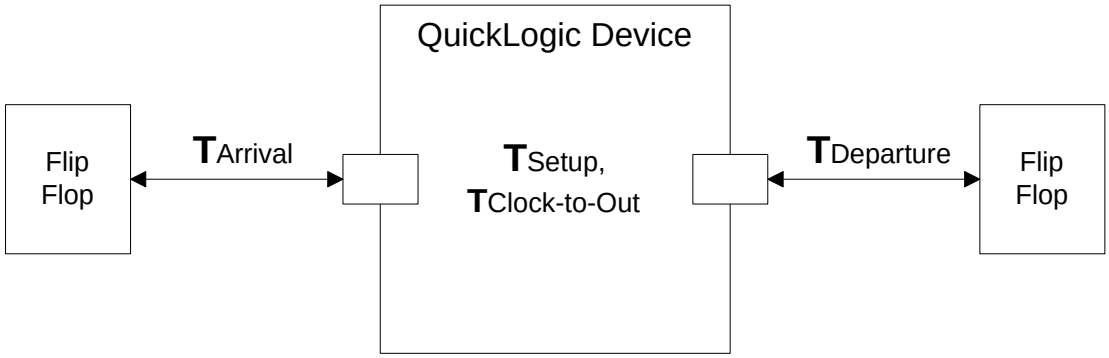


Figure 22-1 describes the following:

```
period = Tarrival + Tsetup = Tclock-to-out + Tdeparture
```

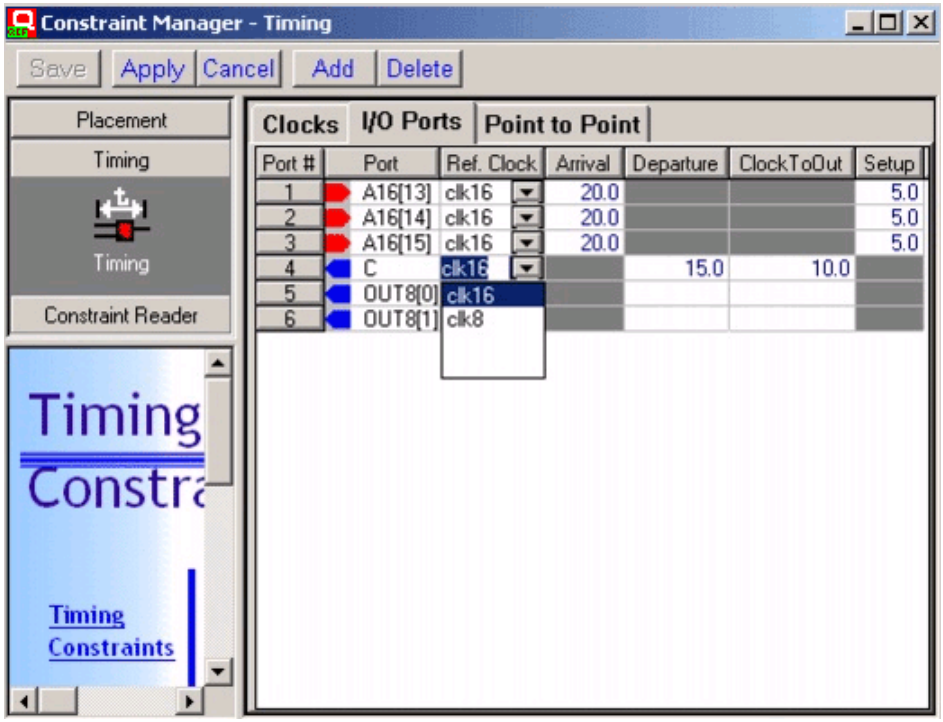
The arrival and setup times have a direct relationship to the period of the reference clock signal. The sum of the arrival and setup times must be equal to the period of the reference clock. If it is not, some flip-flops will have the edge of the clock signal precede the arrival of the data that is supposed to be registered. A similar relationship exists between the Clock-to-Out and Departure times; the sum must be equal to the reference clock period.

With these timing relationships established, you only need to enter two of the three constraints into the editor (define the setup time by entering the clock period and arrival time, or define the departure time by specifying the Clock-to-Out and the reference clock period.) However, it is possible to specify all constraints if the clock period is greater than the sum of the other two constraints. For further information regarding these timing issues see the figures below, or refer to **"Path Analysis" on page 322** for how to calculate setup and clock-to-out times.

© 2008 QuickLogic Corporation

22.1.1.3.5 Reference Clock

After you have all of the I/O ports selected for timing constraints, specify a reference clock signal for each one. If there is only one clock signal for the design, the editor automatically selects that clock signal as the reference. However, if there are multiple clocks you need to choose the proper clock signal. Click in the **Ref. Clock** column next to the appropriate I/O port.



22.1.1.3.6 Setup and Arrival

Specify either the setup and/or the arrival time for any input or bi-directional ports. Keep in mind that you only need to specify one or the other since the reference clock period has already been defined. To enter the timing constraint, click in the Setup and/or Arrival column(s) adjacent to the appropriate I/O Port. Enter a time value (in nanoseconds). Enter the Clock-to-Out and/or Departure time for any output or bi-directional ports, following the same steps outlined above.

22.1.1.3.7 Point-to-Point

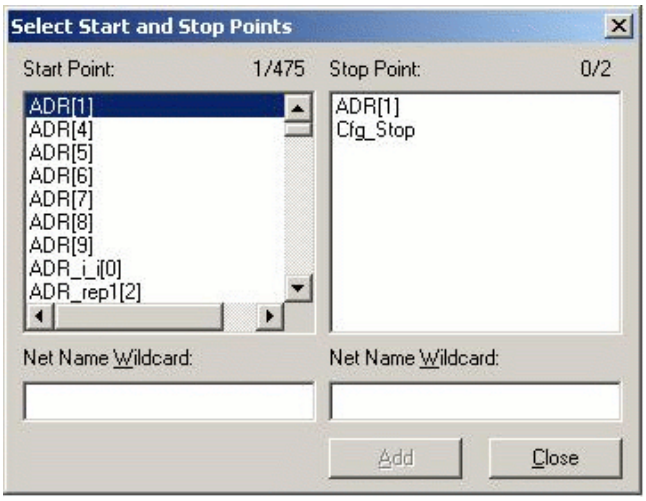
The final timing constraint to specify for the design is the point-to-point constraint. For those familiar with the Path Analyzer, point-to-point constraints refer to any paths that have a starting and stopping point. In fact, any timing constraints entered in the Path Analyzer are automatically used in the point-to-point constraint editor.

To add a point-to-point constraint:

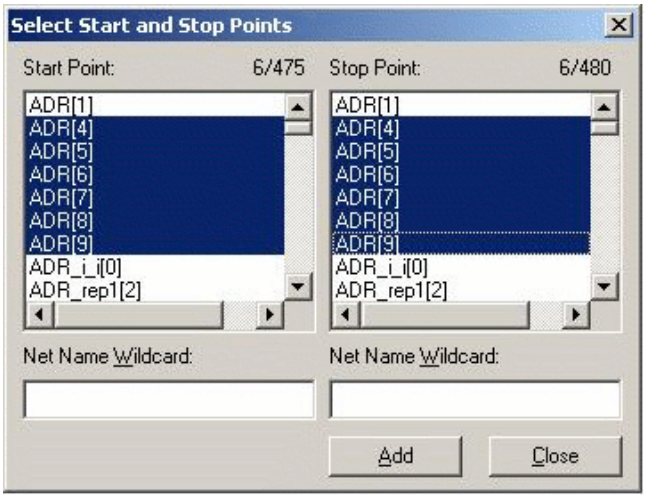
1. In the Timing Constraints dialog box, click on the **Point to Point** tab.

© 2008 QuickLogic Corporation

2. Click **Add**. The Select Start and Stop Points dialog box opens.

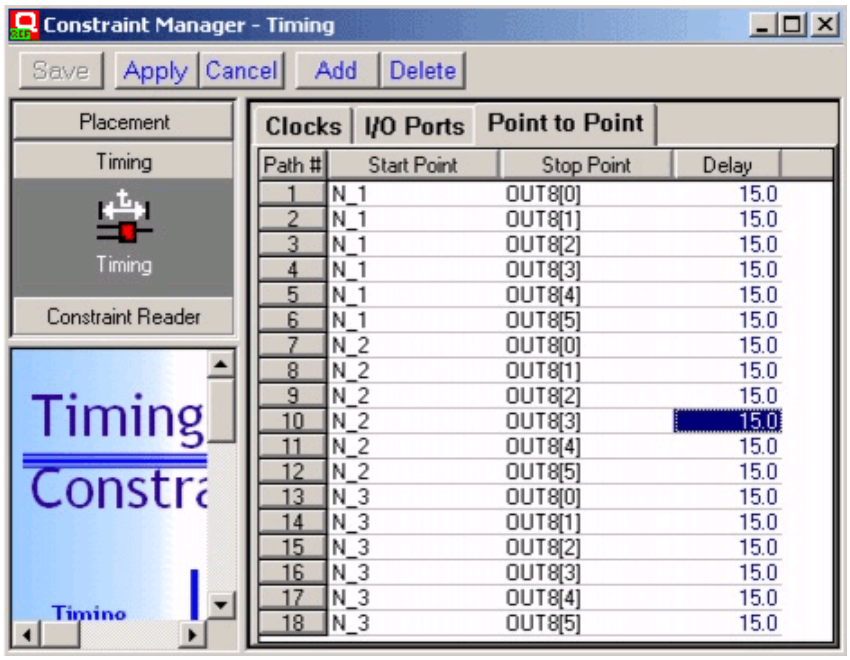


3. From the **Select Start and Stop Points** dialog box, select the paths in the **Start Point** column on the left hand side of the dialog box. The right side of the dialog box lists all of the possible stopping points for your chosen starting point.



4. Select a **Stop Point** and click **Add**. The resulting path is added to the Timing Constraints dialog box.

5. Once you have selected all of the paths, specify the desired point-to-point delay for each path. Click in the **Delay** box and enter a time in nanoseconds.



6. Click **Save** in the Timing Constraints dialog box. All of the timing constraints entered for the design are now saved to a constraint file. From now on every time you will open this design in SpDE, this constraint file will be read, so you won't have to enter the constraint information again.

22.1.1.4 Fan-out Reduction Techniques

QuickWorks version 9.5 and above enables the user to control the fan-out of certain nets—this can be particularly useful to get better performance limited by fan-outs.

The user can control the fan-out of nets by specifying limits on the maximum fan-out allowed on nets—this can be a very useful feature in controlling the fan-out limits.

The designer can use the following commands in .qcf file to control the fan-out of nets:

- **dup_on**—Turn on the fan-out reduction. The default is turn off.
- **high_fan-out_net_limit netname number**—Set the fan-out limit to the specific net. If the net specified has more fan-out than the limit, the fan-out reduction will be applied to the net.

Example:

```
dup_on
high_fanout_net_limit count_a0 5
```

22.1.1.5 Reducing Set-up Time and Clock-to-Out

Utilizing the flip-flops within the I/O pads or utilizing a flip-flop located close to the I/O Pad can reduce set-up time and clock-to-out delays. The Eclipse parts have input and output flip-flops, while the QuickRAM parts have only input flip-flops. The enhancement in QuickWorks

© 2008 QuickLogic Corporation

version 9.5 and above enables the user to put the flip-flops registering the data (input or output) close to the pads. The clock-to-outs are minimized by using the flip-flops within the I/O if it is available (as in the case of Eclipse parts) and by placing the flip-flop close to the I/O pad as in the case of QuickRAM parts. This feature can be used, by specifying which flip-flops need to be pulled into, or close to the I/Os in the .qcf file. The semantics for specifying the flip-flop (FF) movement is:

```
pull_ff_into_io <netname> (FF output)
```

pull_ff_into_io is a keyword, and <*netname*> is a character string, is the net driven by FF. The <*netname*> character string also supports the wildcard (*).

Example:

```
pull_ff_into_io data_reg[*]  //WildCard
pull_ff_into_io CASz
pull_ff_into_io WEz
pull_ff_into_io RASz
pull_ff_into_io CSz
```

This method has provided significant improvements in clock to outs and set up times on benchmark designs.

NOTE: Limitation: the fan-out reduction currently will not work on the nets driven by flip-flops. This limitation will be removed in the subsequent release.

© 2008 QuickLogic Corporation

22.1.2 Placement Constraints

This section explains how to assign placement constraints on pads and flip-flops in your design. Similar to the timing constraints, the placement constraints are written to a constraint file that is read every time the design is opened in SpDE.

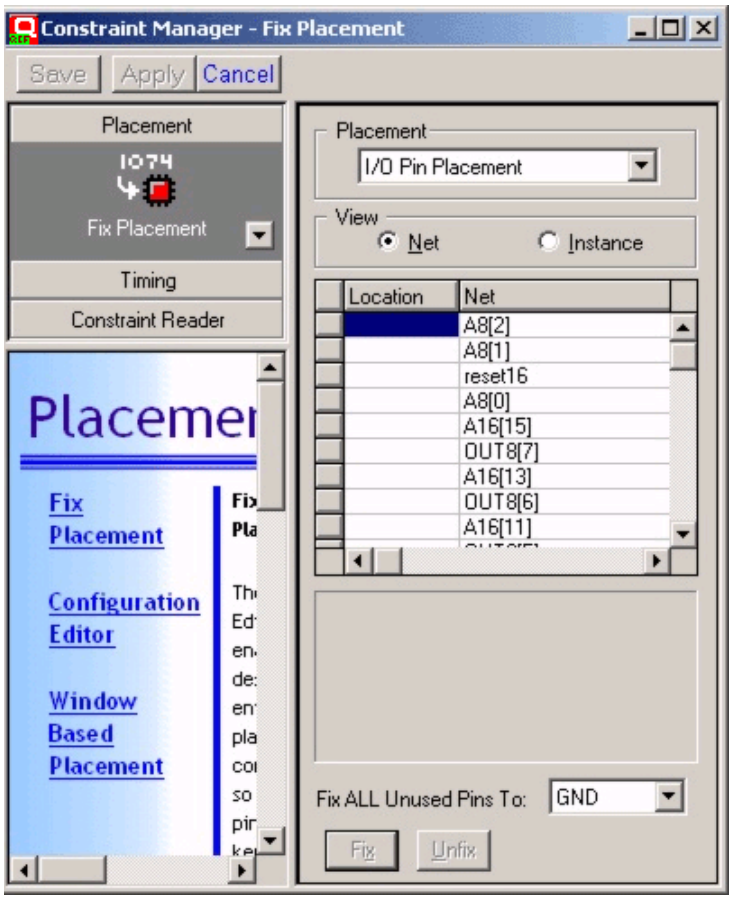
For more information, see also:

- **"Using Constraint Manager" on page 268**

22.1.2.1 Placement Editor

The Placement Editor feature enables designers to enter their placement constraints only once, so the same pin-out can be kept through several design iterations.

To open the Placement Editor, from SpDE, select **Tools>Constraint>Placement>Fix Placement**. The Placement Window opens, where you can enter a fixed placement for all of the pads and flip-flops in the design.



One advantage of entering the placement information from SpDE is that it prevents you from specifying illegal placement information. For instance, SpDE will not allow a clock pad to be placed on a bi-directional pin and vice versa. The Placement Editor allows you to select the

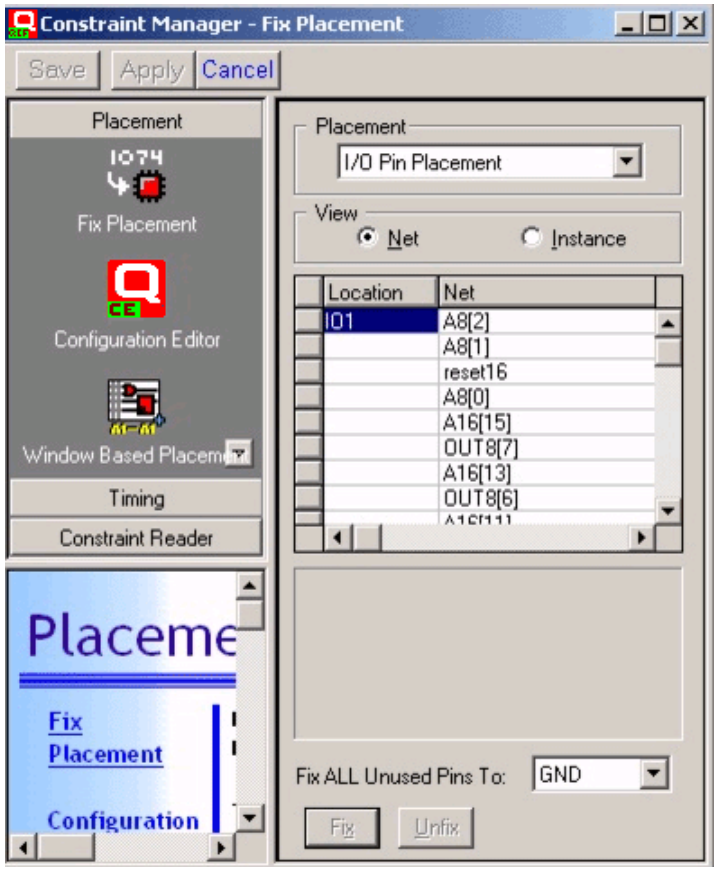
© 2008 QuickLogic Corporation

pads or flip-flops by net name or instance name. To select the pads or flip-flops by net name or instance name, in the View By section of the Placement window, click on the **Net** or **Instance** radio button.

NOTE: The best way to fix placement is to use the net name, not the instance name. This is because synthesis tools may change the instance names of pads or flip-flops during design iterations, while net names stay the same.

To fix the placement of a pad, select I/O Pin Placement in the Placement scroll menu. Locate the net or instance name of an I/O pin and click on the box in the Location column. This allows you to type in a new location.

CAUTION: Do not use the leading I/O with BGA packages.



If you do not follow this format, you will get a message that the specified location is not valid. Once the location has been entered, press **Enter** or click **Fix** on the bottom of the Placement window.

22.1.2.1.1 Fixing the Placement of GPIO Pads

Depending on the design, users may wish to fix place I/Os at different locations. The PolarPro architecture has the following two types I/O pads:

- GPIO (General Purpose I/O)
- DDRIO (Double Data Rate I/O)

As in other architectures, GPIOs can be fix placed on any GPIO cell without any restriction. GPIO gate can also be placed in an unused DDR set. While swapping DQS gate with GPIO gate placed in DDR set, set base swapping will take place.

NOTE: DDRIO can be used as GPIO, so ideally one would fix place GPIO on DDRIO. However, this support is not available in the current release. Therefore, GPIO can only be fix placed in GPIO banks.

22.1.2.1.2 Fixing the Placement of DDRIO Pads

The PolarPro architecture has introduced a new type of I/O pads – DDRIO. There are three different type of DDRIOs:

- DDR-DQ.
- DDR-DQS.
- DDR-Differential Clock

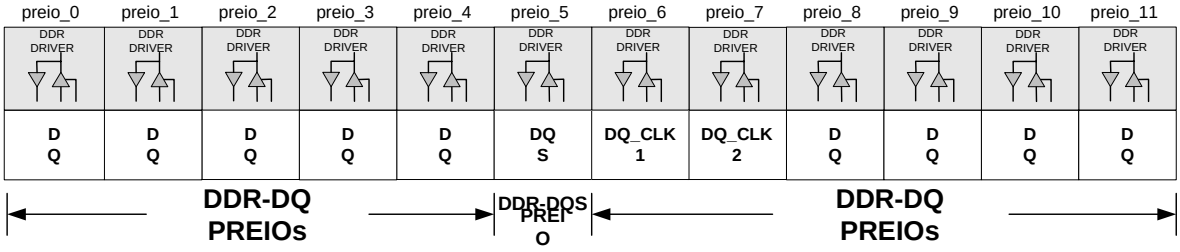
NOTE: A DQS strobe signal used in reading in and writing out data is generated inside the DDR-DQS PREIO. Every set of DDR I/Os will have its own DQS signal to be used with 9 up to 11 DDR-DQ I/Os in the group.

PolarPro reserves one bank for DDR PREIOs. This bank is divided into sets of 12 DDR-PREIOs as follows:

- 9x DDR-DQ PREIOs configured for DQ and DM signals
- 1x DDR-DQS PREIO
- 2x DDR-DQ PREIOs configurable as DQ or differential input or output clocks

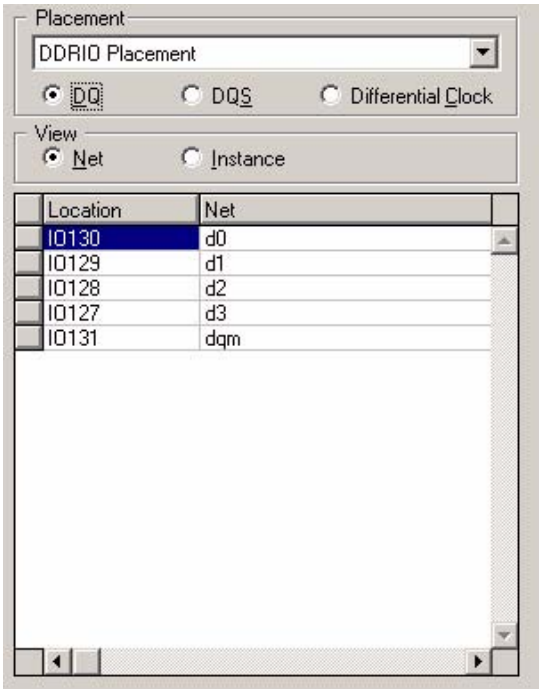
Figure 22-2 shows the DDR PREIO signal arrangement.

Figure 22-2: DDR-PREIOs



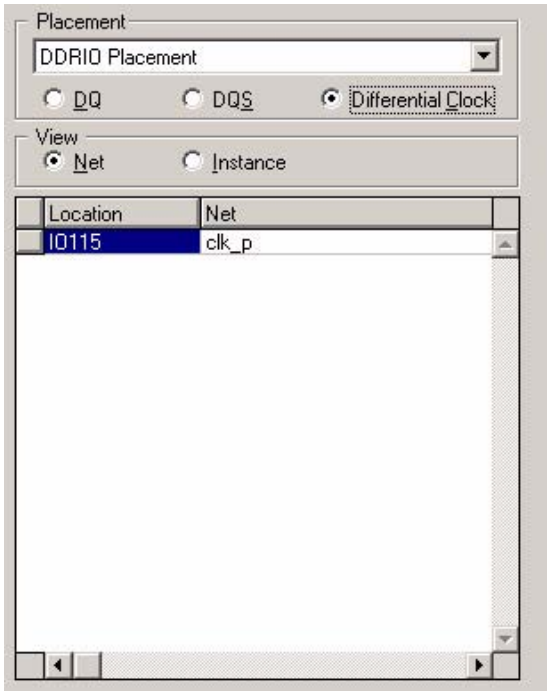
© 2008 QuickLogic Corporation

The following graphical user interface is provided for DDRIO fix placement.



There are three radio buttons in the Placement area to toggle among **DQ**, **DQS** and **Differential Clock**. A location is specified by editing the information under the Location tab or by dragging the net or gate to the desired valid location.

NOTE: Under the Differential Clock setting, all DDRIOs that are used as Differential Clock Plus Gate/Cell are listed. Minus Differential Clock is not listed because plus and minus together form a pair which acts as a single entity.



© 2008 QuickLogic Corporation

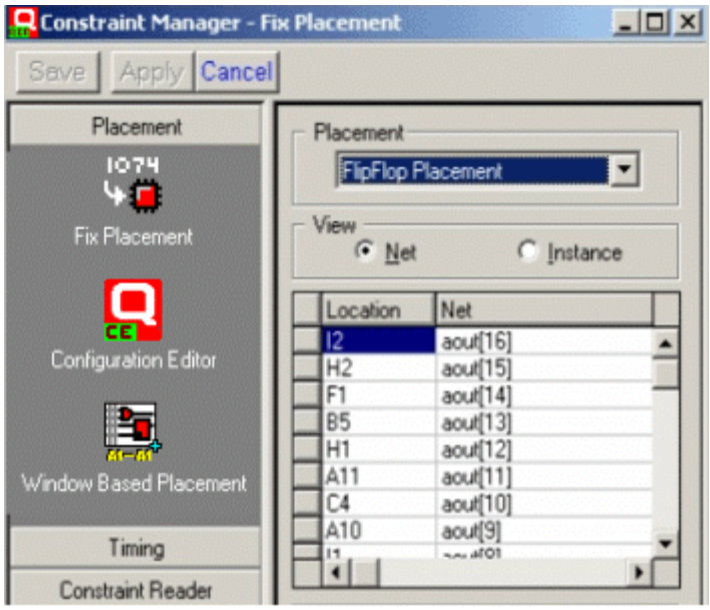
Fix placement of DDRIOs has a few restrictions as follows:

- DDR-Differential Clock I/Os are independent of the rest of the DDR set. They can independently move around in different sets.
- DDR-DQS has a higher priority within a set.
- For DDRIOs (DQS & DQ), only set base movement is allowed via the DQS signal.
- DQ movement is allowed within a set.
- Whenever DQS is moved from one set to another, all the associated DQs are moved to the specified set.

NOTE: SpDE also supports drag-and-drop functionality for fixing pins and flip-flop placements.

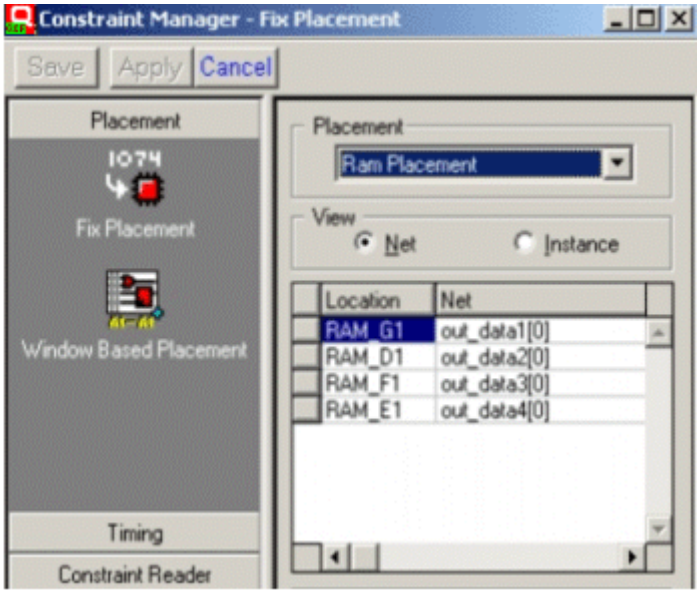
Once changes to the placement information are made, click **Save** to save them to the constraint file. The placement constraints are written to the QuickLogic Constraint file (`.QCF` file extension). These placement constraints will be used every time the design is opened in SpDE.

The same process is used to fix the placement of flip-flops; the only difference is that flip-flop locations are specified in array format according to the row and column of the flip-flop (e.g., location B3 is the second column, and third row of the device). To view a list of all the flip-flops in the design, from the Placement section of the Placement window select Flip-Flop Placement in the Placement scroll menu.



© 2008 QuickLogic Corporation

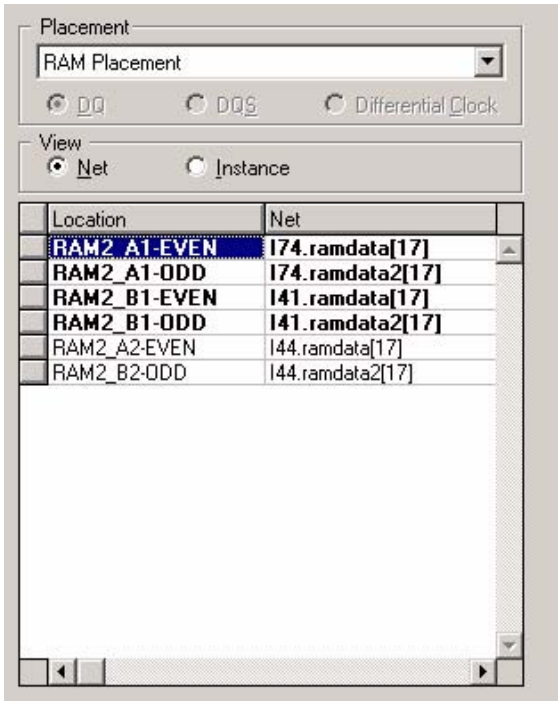
Embedded Dual-Port RAM blocks are only in QuickRAM, QuickPCI, Eclipse, and PolarPro devices. The fix placement of RAM blocks feature is only available for QuickRAM, QuickPCI, Eclipse, and PolarPro devices. The same process is used to fix the placement of RAM blocks. To view a list of all the RAM blocks in the design, select **Ram Placement** from the Placement drop-down list box.



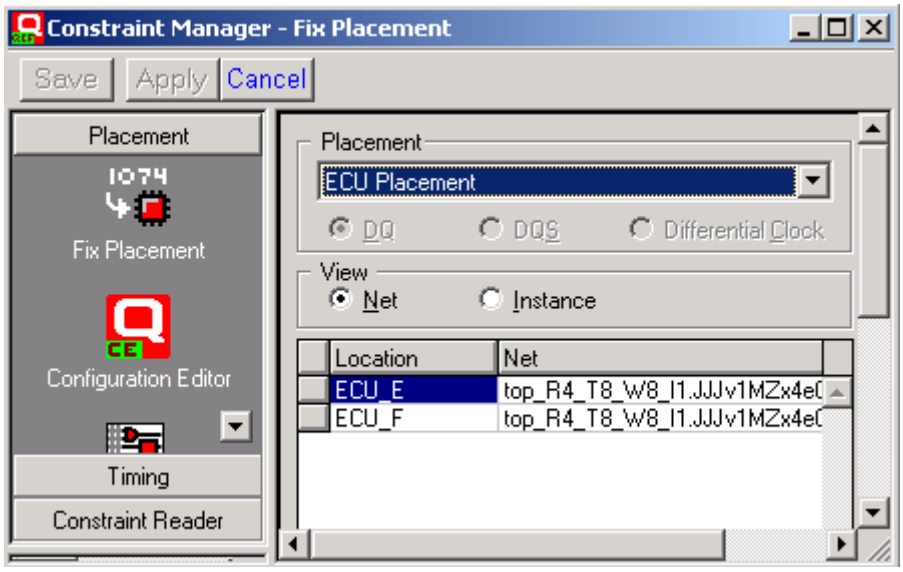
The PolarPro architecture supports the following two RAM Gate types:

- 4 K RAM Gate
- 8 K RAM Gate

Each RAM cell in PolarPro devices is 8 K and is implemented using a 2x1 RAM block, each consisting of a 4 K RAM block. For more information on usage, refer to the QuickLogic Application Note 86 at http://quicklogic.com/images/appnote86.pdf. A 4 K RAM gate can be placed on even or odd parts of any RAM cell, or two different 4 K RAM gates can be placed in single RAM cell.

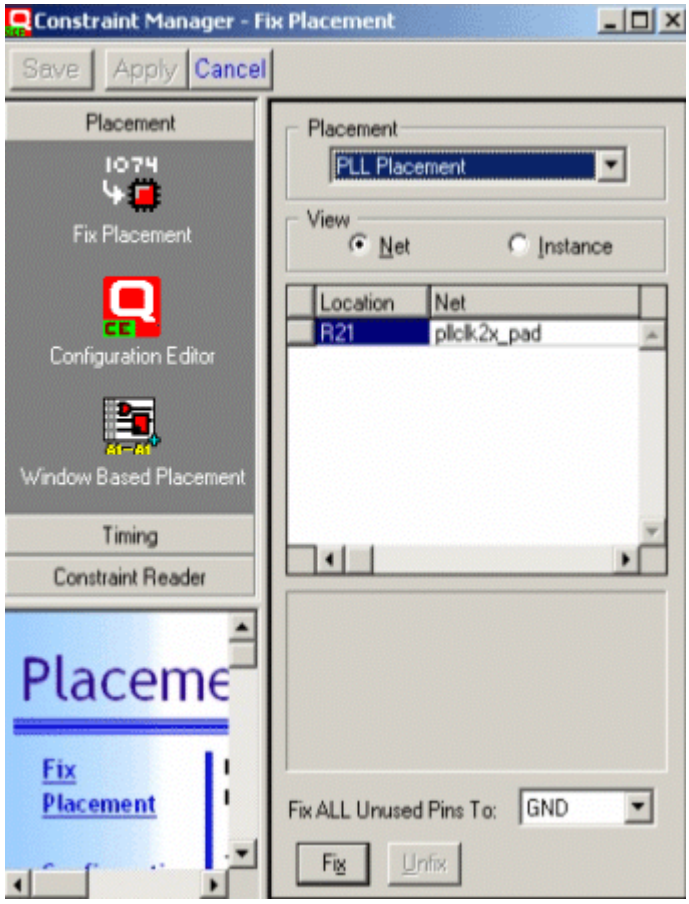


To view a list of all the Embedded Computational Units (ECUs) in the design, select **ECU Placement** from the Placement drop-down list box.



© 2008 QuickLogic Corporation

Phase Lock Loops (PLLs) are available in Eclipse devices.The ability to perform fix placement of PLLs is only available for Eclipse devices. To view a list of all the PLLs in the design, select **PLL Placement** from the Placement drop-down list box.



For more information, see also:

- **"Using Constraint Manager" on page 268**

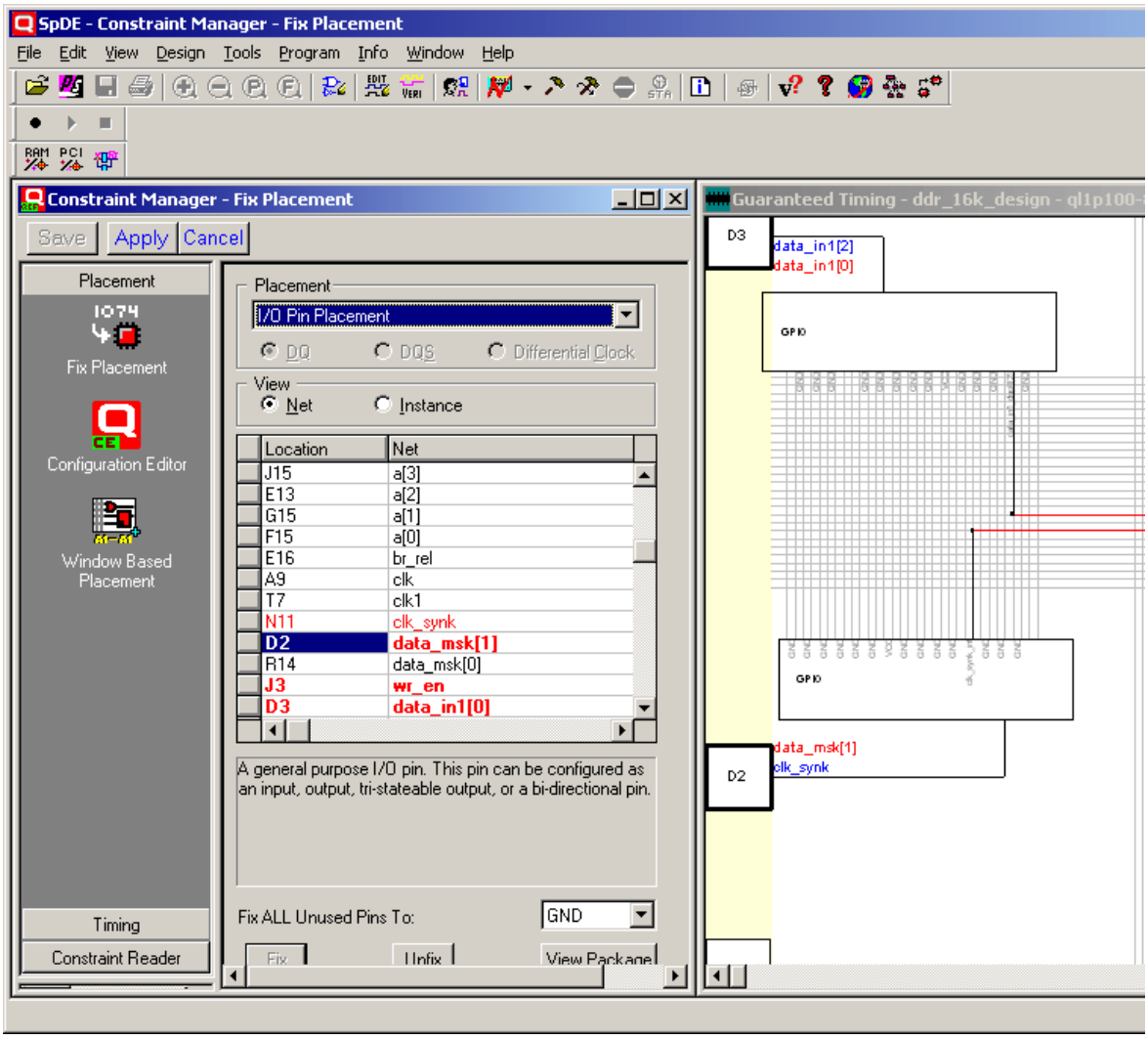
22.2 Drag-and-Drop Functionality

One of the most useful features of the Placement Editor is support for drag-and-drop functionality from the Physical Viewer and/or Package View to the editor window and vice versa.

This operation is very similar to the drag-and-drop functionality of the Windows Explorer file manager system.

To use the drag-and-drop functionality:

1. From the Placement Editor, click on either a pin location or a net name, leaving the mouse button depressed.
2. Drag the mouse cursor until it is directly over the pin in the Physical Viewer, where you want to fix the net.
3. Release the mouse button and the new pin placement is reflected in both the Physical Viewer and the Placement Editor by placing the Net name to the new pin.
4. To save the file in the Placement Editor, click on **Save** to have the new pin placement be in effect.

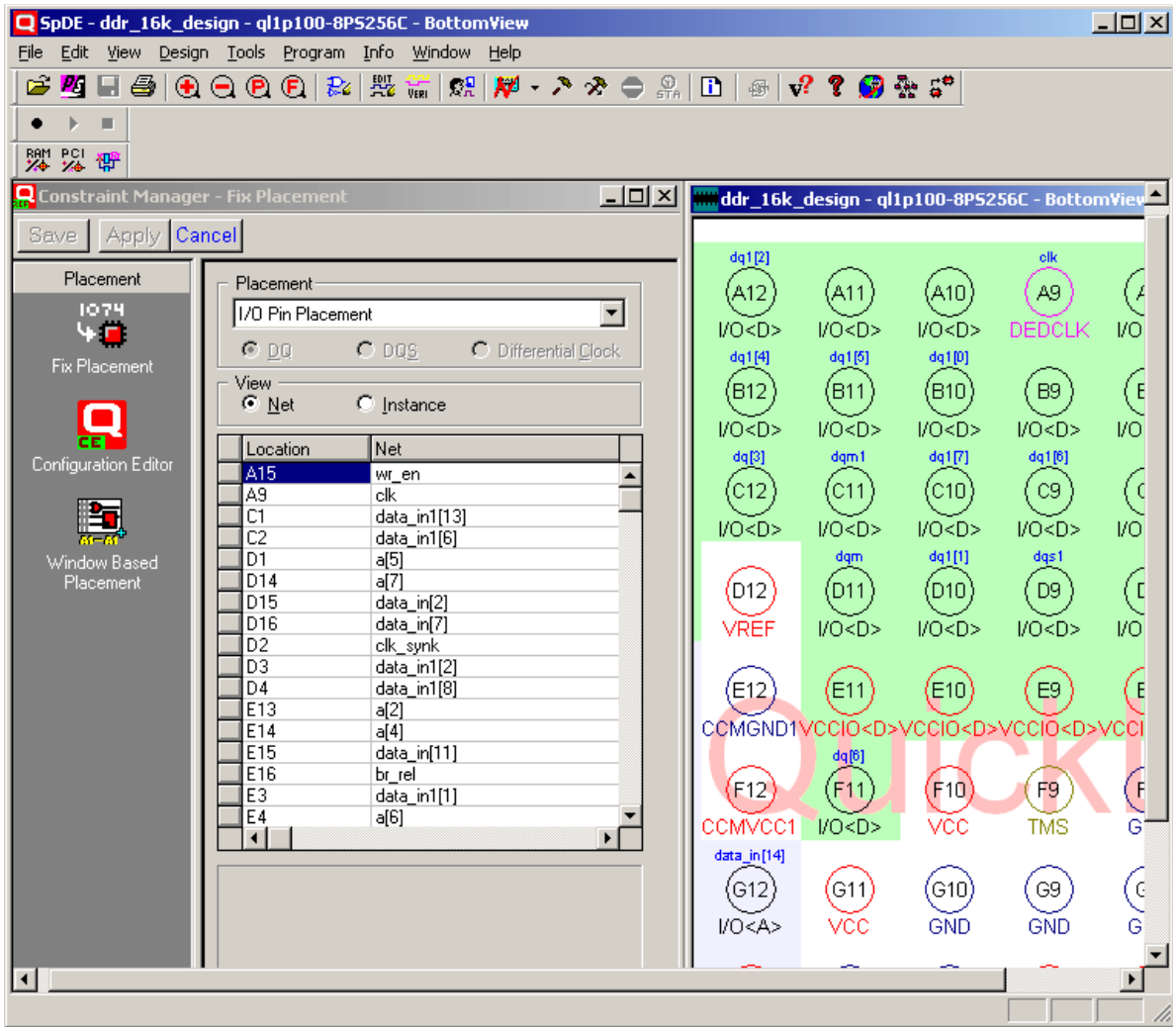


© 2008 QuickLogic Corporation

Pins can also be fixed using the Package View:

1. To view the design in Package View, select **View>Package View**. This operation is very similar to fixing the pin locations from the Physical Viewer in SpDE.
2. From the Placement Editor, click on either a pin location or a net name, leaving the mouse button depressed.
3. Drag the mouse cursor until it is directly over the pin in the Package View where you want to fix the net.
4. Release the mouse button and the new pin placement is reflected in both the Package View and the Placement Editor by placing the Net name to the new pin.
5. To save the file in the Placement Editor, click on **Save** to have the new pin placement be in effect. Fixing flip-flops, RAM blocks, PLLs, and ECUs are not possible in Package View because none of the internal characteristics of the designs are visible.

NOTE: Pin placements can be fixed when SpDE is in either the Physical View or Package View. Flip-flop placements can be fixed only when SpDE is in the Physical View; Flip-flops placement cannot be fixed when SpDE is in the Package View mode.



22.3 Fix Placement Using the QCF File

Another approach to fixing the placement of I/Os, flip-flops, RAM blocks, ECUs, and PLLs is by editing the `.qcf` file. If a `.qcf` file does not already exist, you can create one using a text editor and include `.qcf` file extension with the file name.

NOTE: Only edit the .qcf file while SpDE is NOT RUNNING. Since SpDE modifies the contents of the `.qcf` file, any edits you make while SpDE is running may be lost

To fix a flip-flop in the `.qcf` file, enter:

```
ffplacement <<FF output signal>> <<logic cell physical location>> [FF Location]
```

where:

FF Location can be either F1 for QZ or F2 for Q2Z. If unspecified, default is F1.

Example:

```
ffplacement    N_8 BI56 F1
ffplacement    N_53 BG44 F2
```

The ffplacement constraint specifies to fix the net N_8 to Flip Flop1 (port QZ) of logic cell BI56 and the net N_53 to Flip Flop2 (port Q2Z) of logic cell BG44 respectively.

NOTE: The ffplacement format is backward compatible meaning that the previous versions of constraint files <design.qcf> do not need any changes. The ffplacement keyword must be lower-case.

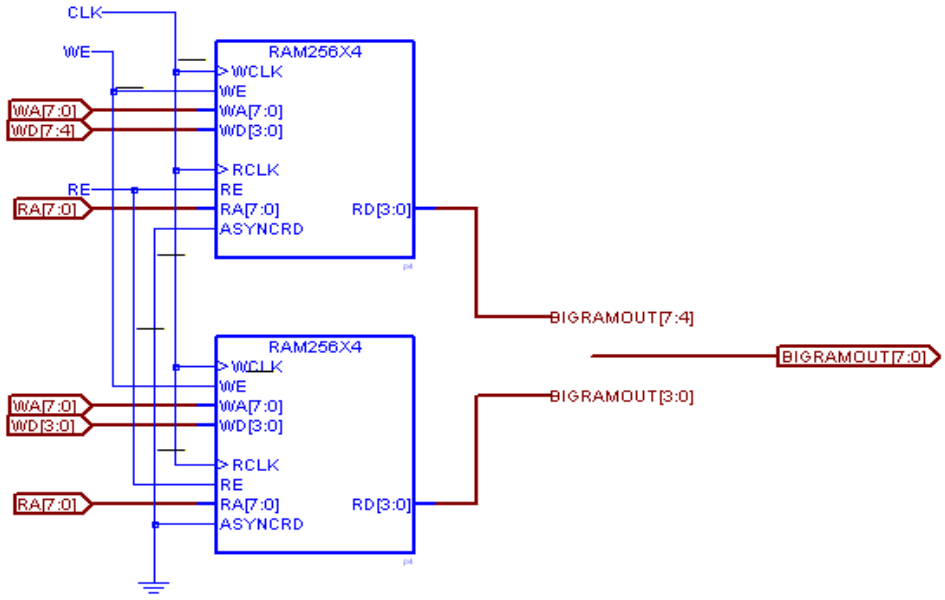
To fix a RAM block in the `.qcf` file, enter:

```
placeram <<read data output signal>> <<RAM physical location>>
```

The placeram keyword must be lower-case. The read data output signal name should match the case used in the design.

In the example shown in **Figure 22-3**, two 256 x 4 RAM blocks made up a larger 256 x 8 RAM. The TOP LEFT corner of the device will be fixed.

Figure 22-3: Example of a multi-block RAM



© 2008 QuickLogic Corporation

Figure 22-3 on page 306 illustrates a graphical implementation of the larger RAM. In this RAM, the read data outputs are called `BIGRAMOUT[7:0]`. Bits 7 to 4 come from the first RAM block and bits 3 to 0 come from the second RAM block.

To fix the placement of these two blocks to the top-left corner of the device, add the two following lines to the `.QCF` file:

```
placeram BIGRAMOUT[7] RAM_A1
placeram BIGRAMOUT[3] RAM_B1
```

The signal name used after the keyword `placeram` is any signal coming out of the target RAM block.

The RAM physical locations correspond to the two ROWS of RAM blocks. They are:

```
RAM_A1, RAM_B1, RAM_C1 etc. at the TOP (the 1 row)
RAM_A2, RAM_B2, RAM_C2, etc. at the BOTTOM (the 2 row)
```

To fix the same RAM to the bottom left corner of the device, the `.QCF` file would read as follows:

```
placeram BIGRAMOUT[7] RAM_A2
placeram BIGRAMOUT[3] RAM_B1
```

To fix an Embedded Computational Unit (ECU) in the `.QCF` file, enter:

```
placeecu <<output signal>> <<ECU physical location>>
```

To fix a Phase Lock Loop (PLL) in the .QCF file, enter:

```
place <<PLLCLK_IN input signal>> <<pin physical location>>
```

For more information, see also:

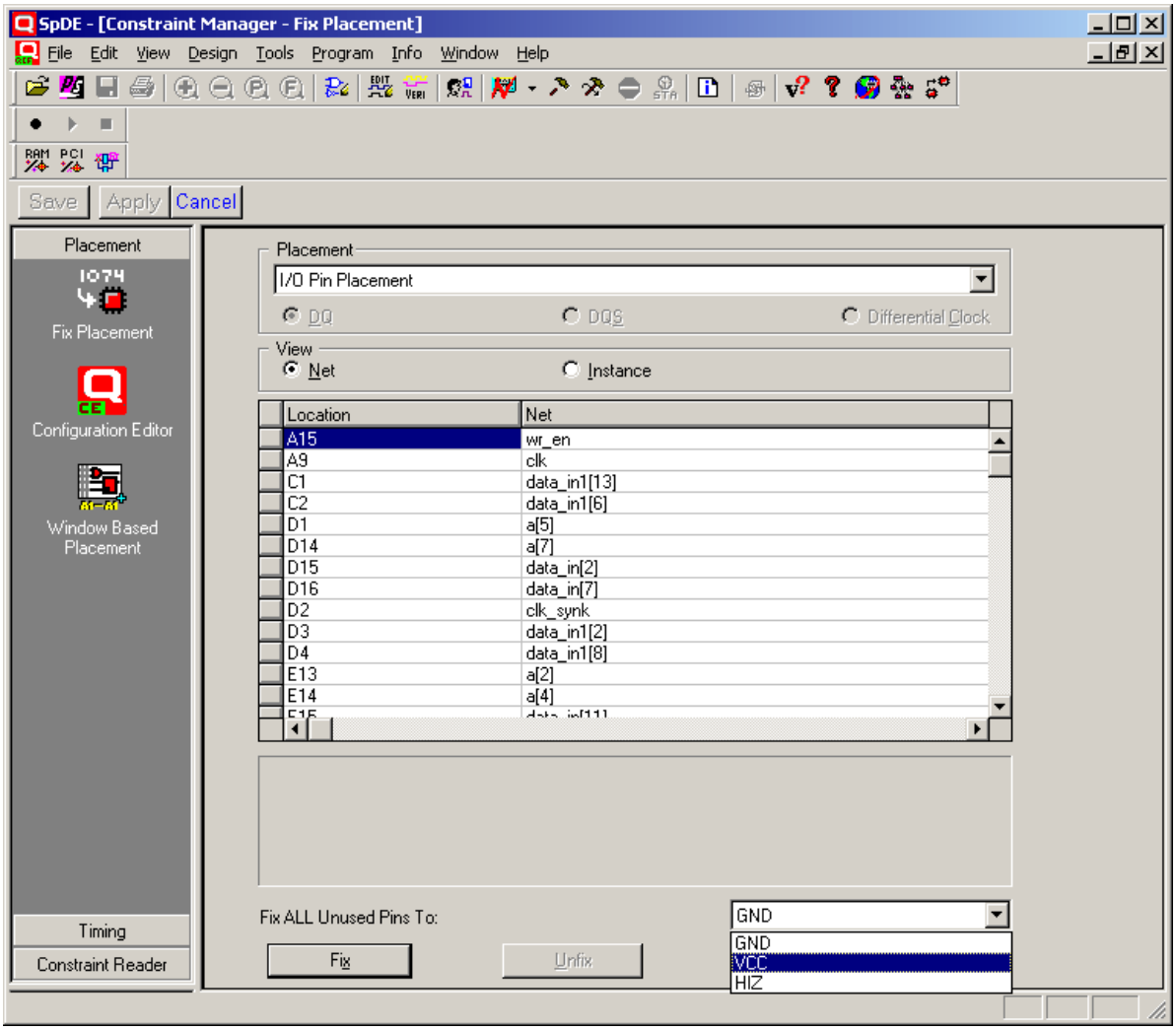
- Constraint File Format (.QCF File) [PDF]
- **"Using Constraint Manager" on page 268**

22.4 Fixing Unused Pins

When a device is programmed, the unused pins of the design are tied off to a selected value. To select the value, from the **Fix ALL Unused Pins To** drop-down list, select **GND**, **VCC**, or **HiZ**. Click **Save** to save this setting to the constraint file (`.QCF`).

NOTE: Once these placement and timing constraints are saved, the tools use this information when you run the place and route tools.

Previously, if any value besides ground was desired for unused pins, the designer had to manually tie the pins off to supply voltage or instantiate TRIPADs in the design for HiZ.



NOTE: To minimize power consumption, fix the unused I/O pins to GND (default) or VCC. If you fix the unused I/O pins to Hi-Z, the power used by the device may increase, unless the pins are pulled to GND or VCC external to the device.

For more information, *see* also:

- **"Using Constraint Manager" on page 268**

© 2008 QuickLogic Corporation

22.4.1 Changing the Sorting Order

Initially, the entries displayed in the pin editor are sorted in ascending order by location names. You can click on the **Location** title box to change the sorting order. Click it again to reverse the order. To aid the user in selecting a particular net or instance by its name, it is also possible to sort the names alphabetically. Click on the **Net** title box to sort the Net names. Click it again to reverse the order of names. In the same way you may also click the **Instance** title box to sort the instance names.

22.4.2 Panning to an Item

To pan to a pin, flip-flop, or a net in a physical view double-click the corresponding box in the Pin Editor. The physical view automatically pans to bring the selected item into the visible region.

22.4.3 Verify Timing Constraints

There are two ways of verifying that the placed and routed design has met the timing constraints specified in the Constraint Editor:

- Run the Path Analyzer for the point-to-point constraints.
  For the proper techniques, refer to **"Path Analysis" on page 322**.
- View the report file to see other timing information (setup and clock to out times).

22.5 Highlight Net and Static Timing Analyzer

After a design has been placed and routed, it is often necessary to carefully analyze timing paths to determine the speed of the design. SpDE contains two tools for this purpose:

- Highlight Net
- Static Timing Analyzer

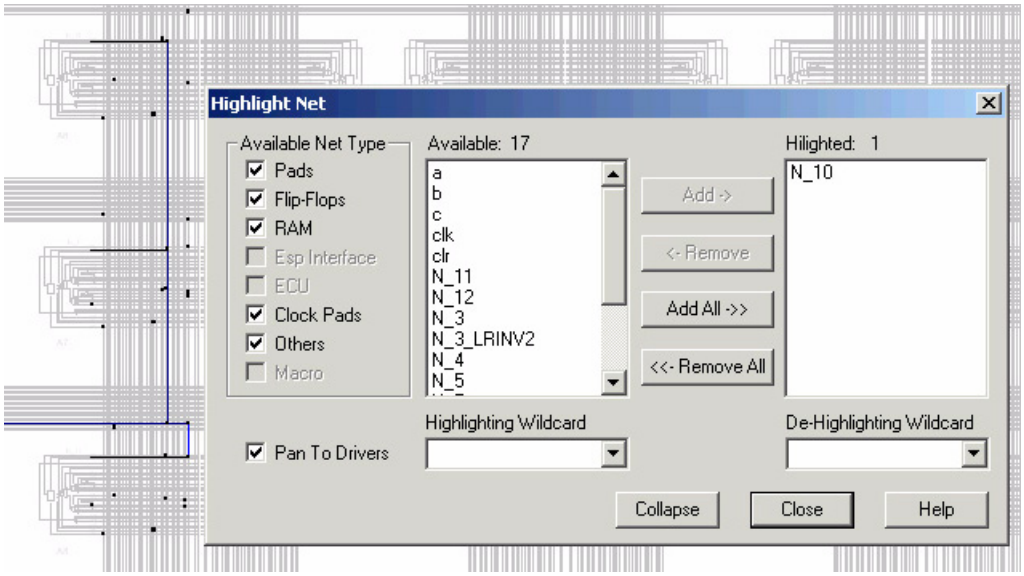
© 2008 QuickLogic Corporation

22.6 Highlight a Net

The Highlight Net mode allows for different methods of analyzing a design by highlighting and de-highlighting nets in SpDE's Physical View. Highlight Net mode is disabled until a design has been placed and routed.

To open the Highlight Net dialog box in **SpDE**:

1. Select **View>Highlight Net**. The Available column lists all the available nets for the selected type in the design.



2. Double-click on a net name in the Available column to move it to the Highlighted column on the right. The net will be highlighted in the physical view of the chip.
3. Double-click on a net name in the Highlighted column to de-highlight the net in the physical view, and to move it back into the Available column.

   NOTE: Alternatively, to move nets between the Available and Highlighted lists, select **Add->** and **<-Remove**.

4. To move the net name from the Available to the Highlighted box, while in Highlight Net mode, click on a wire in the physical view to highlight the entire net containing that wire. Click on a highlighted wire in the physical view to de-highlight the net.
5. If you wish to remain in Highlight Net mode and have the maximum viewing area available for the Physical Viewer, click **Collapse**. Only the title bar remains visible. While in this state, you can continue to highlight and de-highlight nets in the Physical View. To return the Highlight Net dialog box to its original size, double-click the title bar.
6. To exit from Highlight Net mode, click **Close**.

22.6.1 Pan to Net Driver

To allow SpDE to automatically pan to the driver of the net that is selected, in the Highlight Net dialog box, select **Pan to Drivers**. SpDE automatically pans to the driver of the net that is selected whether it is selected by clicking on a wire in the physical view from the available list or if the net name is passed to SpDE from the SCS Hierarchy Navigator for cross-probing.

© 2008 QuickLogic Corporation

22.7 Clock Network Usage

The Clock Network Usage tool highlights all of the nets used on the clock network in the viewer for a given clock input net.

22.7.1 Show Clock Network Usage Tool

This tool can be invoked in SpDE in one of the following two ways:

- From the SpDE menu bar, select **View>Show Clock Network Usage**.
- Press **Alt+V+U**.

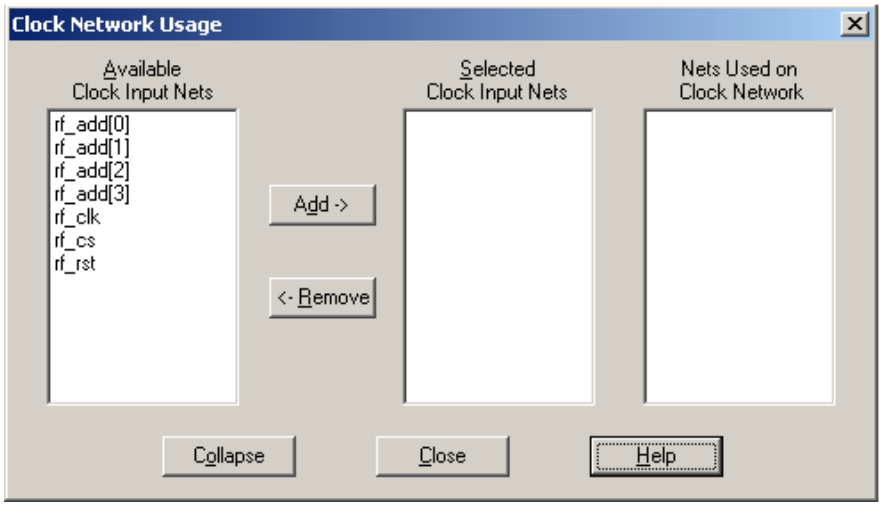
The tool is not available in the following two instances:

- When the Router tool has not been run
- When the Static Timing Analyzer is open

The *Show Clock Network Usage* menu item is disabled in both of these cases.

22.7.2 Using the Show Clock Network Usage Tool

The following dialog box appears when you invoke the tool as explained in **"Show Clock Network Usage Tool" on page 311**.



Clock input nets available in the loaded design are displayed in the Available Clock Input Nets list box.

1. In the Available Clock Input Nets list box, select a clock net for which you want to highlight the network.
2. Click on the **Add** button.
   The clock net selected are moved to the Selected Clock Input Nets list box. All nets used by the clock network are displayed in the Nets Used on Clock Network list box.
3. Select a net in Nets used on Clock Network list box.
   The net are highlighted in the viewer and the view adjusts to show the driver port of the net.
4. To have a better view of the physical viewer, click on the **Collapse** button. The dialog box collapses to a bar. To expand the dialog box, double-click on the bar.

© 2008 QuickLogic Corporation

22.8 Report File

The report file can be generated in the following situation:

- **Default Report**—After the running any or all of the tools, a report is generated by default. This can also be generated from the menu option **Info>Reporting>Report File**.

22.8.1 Default Report

The report file is generated by default after running the tools or by choosing the menu option **Info>Reporting>Report File**. It is generated in the HTML format with headers of the report file contents displayed as hyperlinks at the beginning of the file. The following are the contents reported by default:

- Design Information
- Operating Conditions
- Utilization Information
- Clock Network Utilization by clock pads
- Clock Network Utilization by Internal Logic
- Clock Network Utilization by PLL
- Available HSCK/QMUX Clock Networks/QMUX
- Clock Network Load Information
- Tools run on design `<design name>`
- Pin Table
- I/O Banks, Bank-I/Os info and their VCCIO, VRef values
- Fixed Flip-Flops
- Fixed RAM cells
- Fixed ECU cells
- Nets Removed by Technology Mapper
- Clock Signal Information
- Flip-flops clocked by non-clock cells

22.8.1.1 Design Information

This section provides information about the design and the device used for mapping (*see* **Table 22-1**).

Table 22-1: Design Information

Item	Description
Design	The name of the design loaded into SpDE
SpDE Version	The version of the SpDE tool used to generate the report
Report Generated	The time when the report started generating
CHIP Last Updated	The time when the chip file was last saved
Part Type	The QuickLogic device used for mapping the loaded design
Speed Grade	The speed grade of the QuickLogic device used

© 2008 QuickLogic Corporation

Table 22-1: Design Information (Continued)

Item	Description
Operating Range	This reflects the field temperature range where the device is used like commercial, military, industrial or custom
Package Type	The package used for the device
Link Check Sum	For a sequenced chip file, reports about uniqueness of link list

22.8.1.2 Operating Conditions

The operating voltage, temperature and k-factor are provided in this section (see **Table 22-2**).

Table 22-2: Operating Conditions

Item	Description
Voltage	The operating voltage of the device
Temperature	The temperature at which the device is working
K-Factors	It's a multiplication factor for scaling delay data and it is dependent on speed grade (Kp), voltage (Kv) and temperature (Kt)

22.8.1.3 Utilization Information

This section reports the various resources utilization when the design is mapped to the targeted device. The utilization ranges from logic to programming links within the device. At the end of each line, the percentage of used against the total number of available resources would also be reported (see **Table 22-3**).

Table 22-3: Utilization Information

Item	Description
Utilized cells (preplacement)	The total number of logic cells utilized out of available before placement
Utilized cells (postplacement)	The total number of logic cells utilized out of available after placement
Utilized Logic cell Frags (preplacement)	The total number of logic fragments used out of available before placement
Utilized Logic cell Frags (postplacement)	The total number of logic fragments used out of available after placement
Utilized Fragment A	Total number of logic A fragments used out of available
Utilized Fragment F	Total number of logic F fragments used out of available
Utilized Fragment O	Total number of logic O fragments used out of available
Utilized Fragment N	Total number of logic N fragments used out of available
IO control cells	Total number of I/O control cells used by the design out of available
Clock only cells	Total number of Clock cells used by the design out of available
Bi directional cells	Total number of bidir cells used by the design out of available
RAM cells	Total number of RAM cells used by the design out of available
ECU cells	Total number of ECU cells used by the design out of available
PLL cells	Total number of PLL cells used by the design out of available

© 2008 QuickLogic Corporation

Table 22-3: Utilization Information (Continued)

Item	Description
Flip-Flop of IO cells	Total number of FFs used within the IO cells out of available
1st Flip-Flop of Logic cells	Total number of first FFs used within logic cells used out of available
2nd Flip-Flop of Logic cells	Total number of 2nd FFs used within logic cells used out of available
Routing resources	Total number of routing resources used by the design out of available
ViaLink resources	Total number of link resources used by the design out of available
Utilized Fragment F	Total number of logic F fragments used out of available
Utilized Fragment C	Total number of logic C fragments used out of available
Utilized Fragment T	Total number of logic T fragments used out of available
GPIO cells	Total number of GPIO cells used by the design out of available
GPIO of DDRDQ cells	Total number of DDRDQ cells used as GPIO by the design out of available
GPIO of DDRDQS cells	Total number of DDRDQS cells used as GPIO by the design out of available
DDRDQ cells	Total number of DDRDQ cells used by the design out of available
DDRDQS cells	Total number of DDRDQS cells used by the design out of available
Utilized Fragment F	Total number of logic F fragments used out of available
Utilized Fragment C	Total number of logic C fragments used out of available

22.8.1.4 Clock Network Utilization by Clock Pads

If the Global/Hardwired clock/control signals use the PLLMUXs/QMUXs/GMUXs before driving any load in the design, then any such use would be reported (see **Table 22-4**).

Table 22-4: Clock Network Utilization by Clock Pads

Item	Description
Clock Network	PLL/HSCK MUX/QMUX/GMUX name with respect to the quadrant
Net	The net driving the MUX
Pin	The clock pad pin name
Quad	The owning Quadrant name (e.g.,Top Left, Bottom Right)
Load	Total load on this MUX

22.8.1.5 Clock Network Utilization by Internal Logic

If a net uses HSCKMUXs/QMUXs before driving any of the clock loads, then any such use would be reported (see **Table 22-5**).

Table 22-5: Clock Network Utilization by Internal Logic

Item	Description
Clock Network	HSCK MUX/QMUX name with respect to the quadrant
Net	The net driving the MUX

© 2008 QuickLogic Corporation

Table 22-5: Clock Network Utilization by Internal Logic (Continued)

Item	Description
Driver	The driver cell name of the Net
Quad	The owning Quadrant name (e.g., Top Left, Bottom Right)
Load	Total load on this MUX

22.8.1.6 Clock Network Utilization by PLL

If any of the PLLs/CCMs are used in the design, which in turn use PLLMUXs/GMUXs/QMUXs to drive the clock load, then any such use would be reported (see **Table 22-6**).

Table 22-6: Clock Network Utilization by PLL

Item	Description
Clock Network	PLLMUX/GMUX/QMUX name with respect to the quadrant
Net	The net driving the MUX
PLL	The PLL/CCM cell name with respect to the quadrant. If the PLL is in ESP core, then ESP cell name is reported.
Quad	The owning Quadrant name (e.g., Top Left, Bottom Right)
Load	Total load on this MUX

22.8.1.7 Available HSCK/QMUX Clock Networks

The HSCK/QMUX clock networks are available in each quadrant. If any of them are used by the design, then the remaining number of networks is reported for each quadrant out of the total number of HSCK/QMUX clock networks (see **Table 22-7**).

Table 22-7: Available HSCK/QMUX Clock Networks

Item	Description
Quad TOP LEFT	The number of Top Left HSCK muxes/QuadNets used out of total available
Quad TOP RIGHT	The number of Top Right HSCK muxes/QuadNets used out of total available
Quad BOTTOM LEFT	The number of Bottom Left HSCK muxes/QuadNets used out of total available
Quad BOTTOM RIGHT	The number of Bottom Right HSCK muxes/QuadNets used out of total available

© 2008 QuickLogic Corporation

22.8.1.8 Clock Network Load Information

Reports the usage of the clock network in terms of the number of quad-nets (the parts of each of global clock networks inside a quadrant), total load and number of loads on each type (see **Table 22-8**).

Table 22-8: Clock Network Load Information

Item	Description
Clocknet	The net which drives the clock network
Drivercell	The driver cell of the Clocknet
Quadnets	The number of quadnets driven by the Clocknet across all the quadrants
Quadnet_details	The names of all the quadnets
Total Load	Total number of ports driven by the Clocknet
Load_details	The number of individual type of loads driven by the Clocknet

22.8.1.9 Tools Run on Design <design name>

The time taken by each of the following tools with their set attributes is reported here along with the latest version of that tool: Partdef, Design Logic Optimizer, Placer, Router, Delay Modeler, Back Annotation, Verifier, Sequencer and Auto Buffer.

22.8.1.10 Pin Table

Table 22-9 provides information on all pins in the device.

Table 22-9: Pin Table

Item	Description
Pin #	Pin name for QuickRAM, Pinnacle and PolarPro devices and Serial number for Pre-Pinnacle (pASIC3) devices.
Pad Name	The gate name of the pad. If the pad is not used, it would be connected to GND and the same will be shown as NU(GND).
Net Name	Net name connecting to the pad.
PinType	Reports whether a pin type is IOCONTROL, INPUT, OUTPUT, CLKPAD, or BIDIR.
bankName	It is the IO bank to which the pad belongs.
Fixed	Reports whether the IO is fixed or not. If it is fixed, it cannot be fix placed elsewhere.

© 2008 QuickLogic Corporation

22.8.1.11 IO Banks, Bank-I/Os Info and Their VCCIO, VRef Values

If the I/O banks are not configured using the Configuration Editor, then all the I/O banks have the default I/O standard. Values of VCCIO and Vref are set only for configured I/O banks (see **Table 22-10**).

Table 22-10: I/O Banks Information

Item	Description
I/O Banks	The bank name for the pin
I/O Standard	The standard of the pin
VCCIO	The VCC voltage supply for the pin
VREF	Reference voltage for the pin
PIN #	Pin name

22.8.1.12 Fixed Flip Flops

Reports the information about the FFs that are fix-placed according to the user's choice.

22.8.1.13 Fixed RAM Cells

Reports the information about the RAMs that are fix-placed according to the user's choice.

22.8.1.14 Fixed ECU Cells

Reports the information about the ECUs that are fix-placed according to the user's choice.

22.8.1.15 Nets Removed by Technology Mapper

Reports any nets removed by the Technology Mapper during Logic Optimization.

22.8.1.16 Clock Signal Information

The Clock Signal Information section give details on the driver gate, mcell type and clock skew for each clock source.

The clock skew is defined as the difference between the max and min destinations of clock routing (see **Table 22-11**):

Table 22-11: Clock Skew

Item	Description
CLOCK_NAME	The name of the clock signal
MCELL_TYPE	The type of the clock cell used for clock signal connection
DRIVER_GATE	The gate name of the clock cell
CLOCK_SKEW	The worst case skew of the clock signal

© 2008 QuickLogic Corporation

22.8.1.17 Flip-Flops Clocked by Non-Clock Cells

If the design uses any of the nets on the chip to drive the Flip-flops, for each Flip-flop the following information is reported. This is not a timing report but more of a design analysis (see **Table 22-12**).

Table 22-12: Flip-Flops Clocked by Non-clock Cells

Item	Description
FF_INSTANCE	The instance gate name of the Flip-flop
OWNING_CELL	The owning Logic cell name of the Flip-flop
DRIVING_CELL	The driving Logic cell name of the Flip-flop
DRIVING_NET	The driving net of the Flip-flop

© 2008 QuickLogic Corporation

Chapter 23
Static Timing Analyzer

This chapter describes the features of the Static Timing Analyzer. It contains the following sections:

- **“Functional Overview” on page 319**
- **“Using Static Timing Analyzer” on page 320**

23.1 Functional Overview

The Static Timing Analyzer (STA) is a powerful tool used to perform Path Analysis and Timing Analysis. The Path Analyzer tool identifies the critical paths for optimization. It is used to determine the operating frequency, setup, hold times, and clock skew. The Timing Report gives details about all the clocks in the design including their setup and clock-to- out times. The STA tool also allows the designer to view any violations in the design such as constraint or hold time violations.

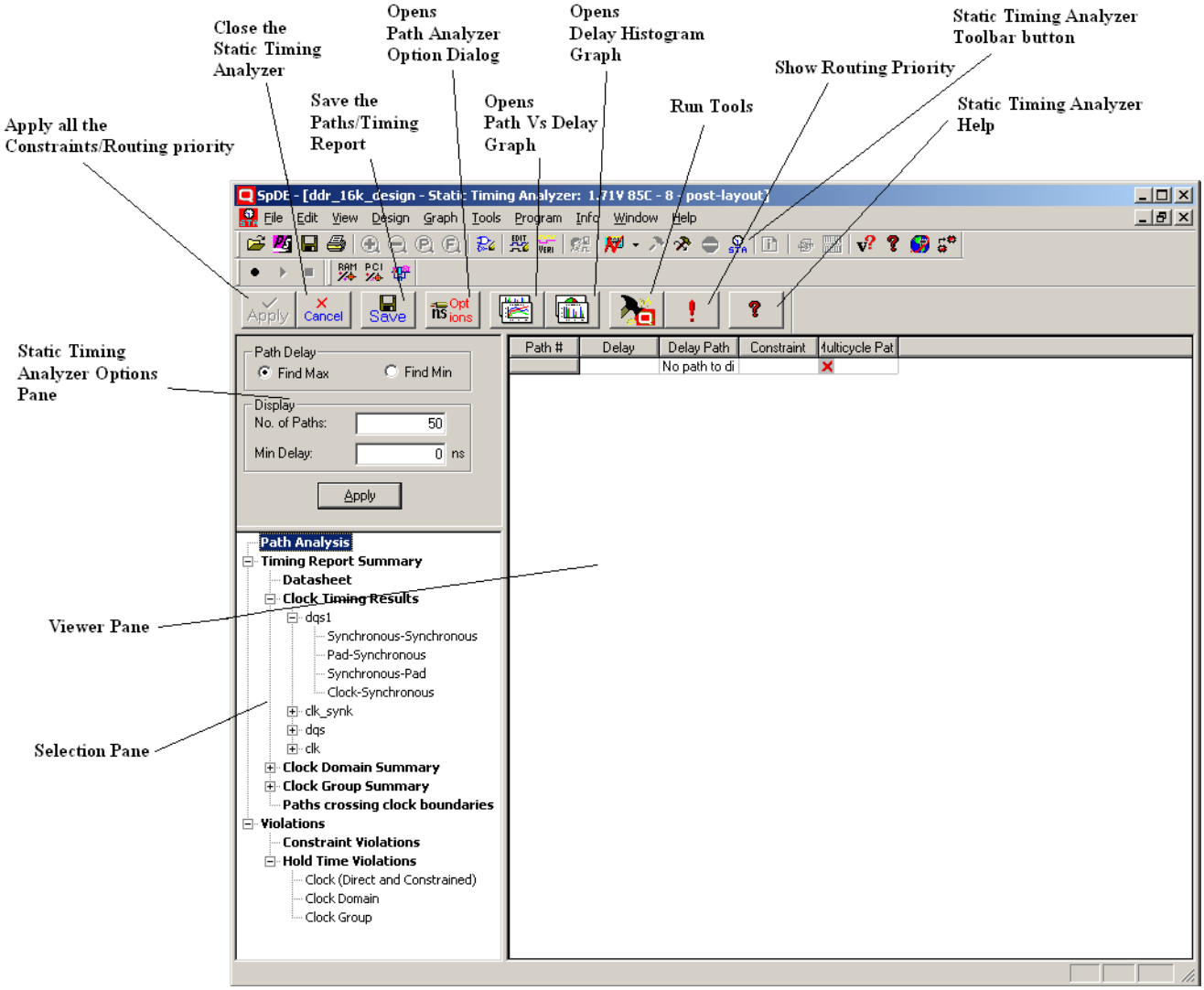
NOTE: The Static Timing Analyzer and the Constraint Manager tools cannot be used simultaneously.

© 2008 QuickLogic Corporation

23.2 Using Static Timing Analyzer

The Static Timing Analyzer provides the designer the ability to perform Path Analysis and Timing Analysis together. It allows the designer to enter constraints, identify critical paths, view a summary of all the clocks in the design, and view all violations. This tool provides an improved user interaction with SpDE.

The Static Timing Analyzer can be accessed using the SpDE toolbar button or by selecting **Tools> Static Timing Analyzer** from the SpDE menu bar.



The Static timing analyzer has the following three panes:

- Static Timing Analyzer Options
- Selection
- Viewer

© 2008 QuickLogic Corporation

23.2.1 Static Timing Analyzer Options Pane

The Static Timing Analyzer Options pane allows users to configure the path delay, number of paths, min delay and max delay.

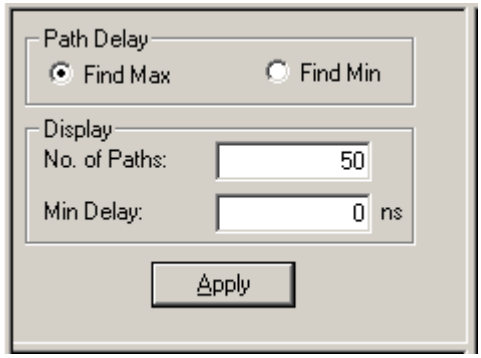


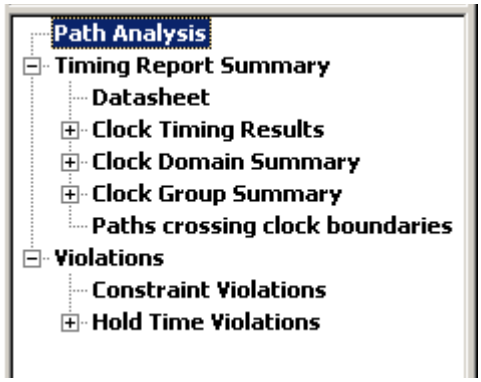
Table 23-1: Static Timing Analyzer Options Pane

Option	Description
Path Delay	The Path Delay radio buttons selects maximum or minimum path delays. Each trail along a given path includes a rising-edge delay and a falling-edge delay. • **Find Max** —the Static Timing Analyzer sums the larger of these edge delays at each trail. • **Find Min** —the Static Timing Analyzer sums the smaller of these edge delays. **NOTE:** This selection does not change the operating conditions (i.e., it does not change worst-case commercial to best-case commercial).
Display	The Display group determines the number of paths calculated and listed in the Static Timing Analyzer viewer pane. The No. of Paths entry limits the number of paths to the specified value. The default is 50 paths, but this can be increased to any size. The Delay entry is interpreted with regard to the Path Delay setting. • **Find Max** —paths will be listed if their delay is greater than or equal to the specified value (Min Delay). • **Find Min** —paths will be listed if their delay is less than or equal to the specified value (Max Delay). Except in special circumstances, this option is rarely used.
Apply	When the designer selects **Find Min** or **Find Max** and changes any Path Delay, No. of Paths, and clicks **Apply** from the Path Analysis pane, the database is updated and the viewer pane shows the information with applied values. If a Path Analysis item is selected from the tree, clicking **Apply** opens the Options dialog to run the Path Analyzer.

© 2008 QuickLogic Corporation

23.2.2 Selection Pane

The Selection pane allows the designers to run the path analyzer, to generate timing report, and to view violations.



23.2.2.1 Path Analysis

The Path Analyzer module is a powerful static timing analyzer used to determine operating frequency, setup and hold times, and clock skew. Working closely with the SpDE Physical Viewer, the Path Analyzer instantly identifies critical paths for optimization. Once the critical path has been identified, the Timing-Driven Placer can be employed to optimize the placement to achieve specified operating constraints.

To run the Path Analyzer, from SpDE select **Tools>Static Timing Analyzer>Path Analysis** or click the **Path Analyzer** icon in STA toolbar. The Path Analyzer Options dialog box opens.

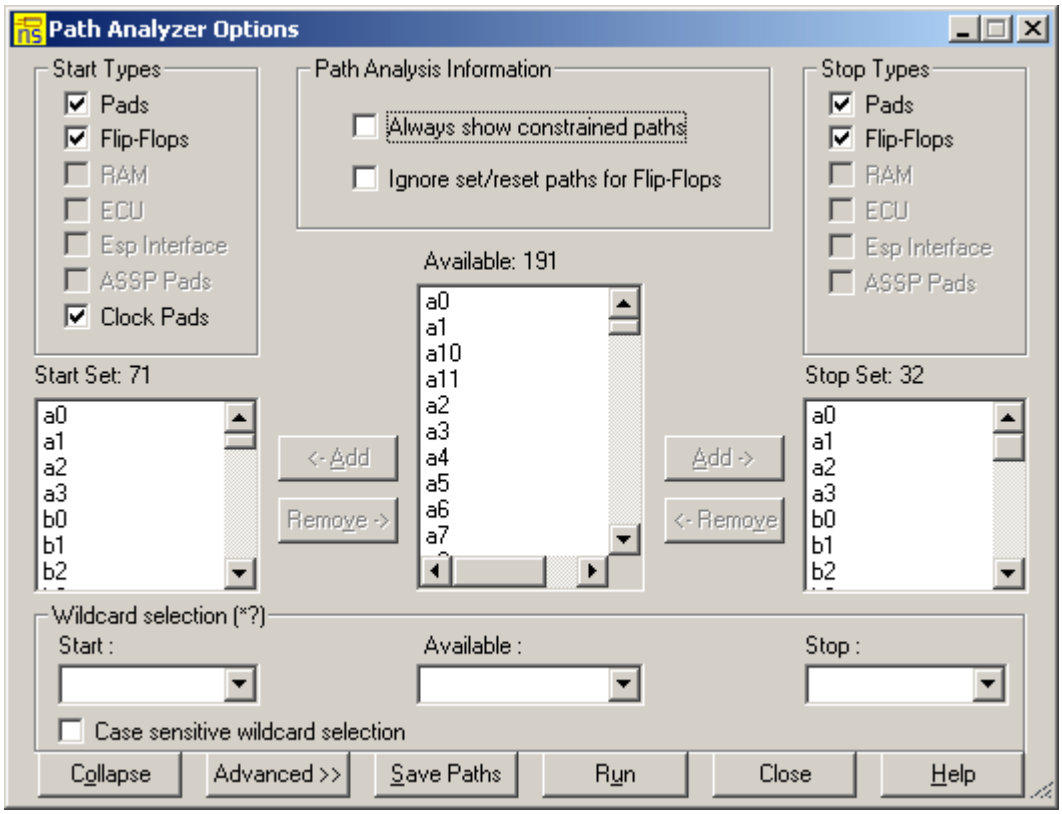
When the Path Analyzer Options dialog is open, select **Run**. The paths will get listed according to the start and stop set selection.

NOTE: The Path Analyzer dialog box has Minimize and Maximize buttons so it can be re-sized as desired.

© 2008 QuickLogic Corporation

23.2.2.1.1 Path Analyzer Options

The Path Analyzer Options dialog box is the primary interface.



The primary purpose of the Path Analyzer Options dialog box is to allow the designer to select the nets in start and stop sets. Once these nets are selected, click **Run**. The Path Analyzer main dialog box reports the delay through paths that start with nets in the Start Set and go to nets in the Stop Set.

To run the Path Analyzer with the newly specified options, click **Run**. *See* **Table 23-1 on page 321** for details on how the Path Analyzer reports delays.

To make the Path Analyzer Options dialog box into a floating title bar, click **Collapse**. The Advanced button is discussed in **"Path Analyzer Scripting" on page 326**.

© 2008 QuickLogic Corporation

23.2.2.1.1.1 Path Analysis Information

Table 23-2 describes the Path Analysis Information options in the Path Analyzer Options dialog box.

Table 23-2: Path Analysis Information Options

Option	Description
Always Show Constrained Paths	To display constraints specified by the user through path analyzer main dialog box, select the **Always show constrained paths checkbox** in the option dialog box. On a successive Run, the path analyzer main dialog box displays all the constrained paths irrespective of their selection in start and stop sets.
Ignore Set/Reset Paths for Flip-Flops	A Flip-Flop may have set/reset input terminals. For many designs, the delay information of paths, which has set/reset terminals in their Start set and Stop set selection, are not very important. Also, for some design there may exist a large list of such paths that may not be user-friendly and easy to read. To ignore all the paths that pass through set/reset terminal of any Flip-Flop, select the **Ignore set/reset paths for Flip-Flops** check box in the Path Analysis Information dialog box. Once the check box is selected, the Path Analyzer main dialog box no longer displays such paths.

23.2.2.1.1.2 Selecting the Start and Stop Sets

The remaining lower sections of the Path Analysis Information dialog box are used to select the Start Set and Stop Set that specify the desired paths. The **Start Set** list box specifies the starting nets for path analysis, while the **Stop Set** list box specifies the ending nets for path analysis. Providing specific Start Set and Stop Set information limits the amount of data in the spreadsheet report, making it easier to interpret the results of the Path Analyzer.

Table 23-3: Selecting the Start and Stop Sets

Option	Description
Start Types	Provides the easiest method for selecting the Start Set list box entries. By default, the check boxes of the available start or stop types should be automatically selected.
Stop Types	Provides the easiest method for selecting the Stop Set list box entries. By default, the check boxes of the available start or stop types should be automatically selected.
Pads	Selects all nets attached to the external terminals of all pads; this check box selects I/O pads and high-drive pads.
Flip-Flops	Selects all nets attached to the output terminals of all flip-flops.
RAM	Selects all nets attached to the pins on the RAM blocks. This option is grayed out if RAM blocks are not available for the targeted device.
ESP Interface	Selects all nets attached to the pins of the ESP block in the device. This option is grayed out if no ESP block is available.
Clock Pads	Selects all nets attached to the external terminals of any pad functioning as a clock (direct and user-defined clocks), not only the internally buffered clocking networks.
ASSP Pads	Selects all nets attached to the external terminals of any pads attached to the pins of the ESP block in the device. This option is grayed out if no ESP block is available.

Selecting one of the Start Set and Stop Set check boxes adds all of the appropriate nets to the desired set. De-selecting one of these check boxes removes all of the appropriate nets to the desired set. For example, assume none of the Start Types check boxes are selected. Selecting the Pads check box adds all pad nets to the Start Set list box. Selecting the Clock Pads check

© 2008 QuickLogic Corporation

box results in no change, since all pad nets are already selected. De-selecting the Clock Pads check box, however, removes the clock pad nets from the Start Set list box, leaving only non-clock pad nets.

23.2.2.1.1.2.1 Select and Move Nets

To select nets manually, use the Available list box in the center of the Path Analysis Information dialog box. Select a **net** or **nets** in this list box and then click on one of the **Add** arrow buttons next to the Available list box. To add the selected nets to the Start Set list box, click the left arrow. To add the selected nets to the Stop Set click the right arrow.

To move the nets out of the Start Set and Stop Set, select a **net** or **nets** and click **Remove**.

To select groups of nets, use the combo buttons below each list box. For example, the bus in [15:0] can be selected by clicking in the Available Wildcard field just below the Available list box and typing ain* or ain [?].

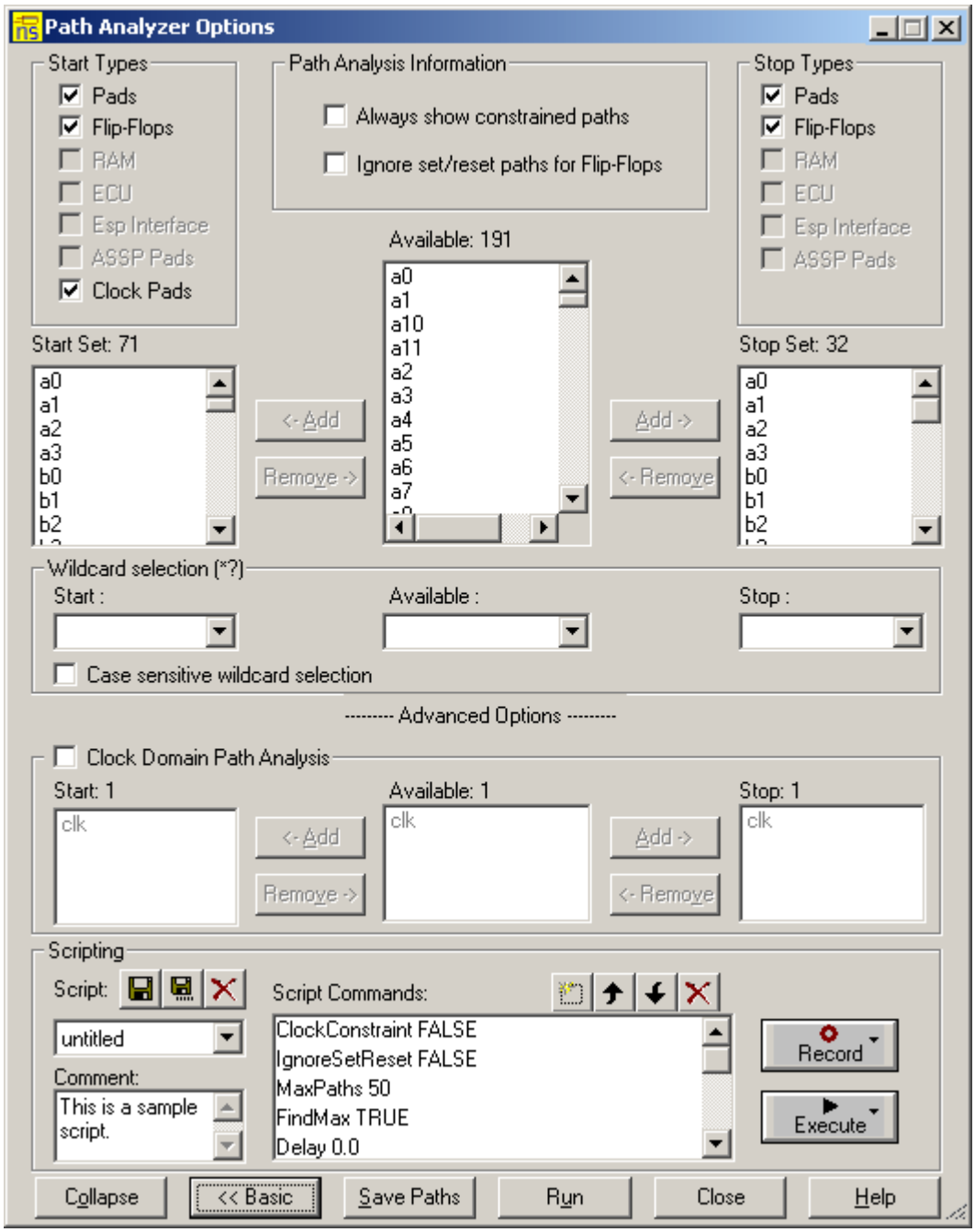
To enter text to selected nets from the Start Set, Available, or Stop Set lists, wildcards can be used. An asterisk (*) represents one or more characters. A question mark (?) represents a single character (e.g., add* selects addr[0], addr[1], addr[2], etc...)

NOTE: Wildcard selections are not case-sensitive. If you want a case-sensitive Wildcard Selection, click the check box with that title. For example, if this box checked, a wildcard of Cl* will not select CLK.

© 2008 QuickLogic Corporation

23.2.2.1.1.3 Path Analyzer Scripting

In the Path Analyzer Options dialog box, the designer can record a script of the button and wildcard entries used in order to easily repeat the selection of a common start and stop set.



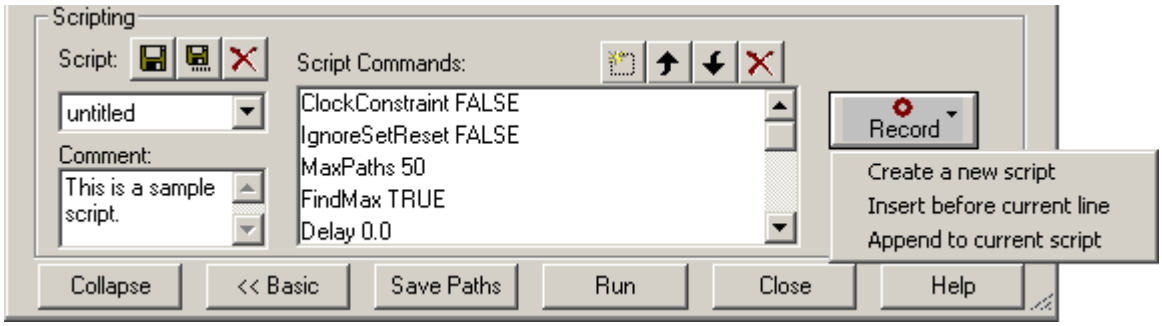
To begin using the Path Analyzer scripting function, in the Path Analyzer Options dialog box click **Advanced**. A sample script window called **untitled** opens.

© 2008 QuickLogic Corporation

23.2.2.1.1.3.1 Record

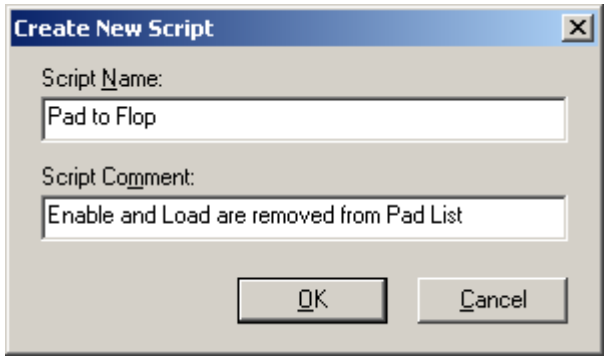
To add a new script:

1. Click **Record**. A Record submenu appears listing three options:
   - **Create a new script**
   - **Insert before current line**—to record additional script commands in the middle of an existing script
   - **Append to current script—**to record additional script commands at the end of a current script.



To create a new script:

1. Select **Create a new script**. A Create New Script dialog box opens, prompting for the script name and comments. Keep the script name short and use only alphanumeric characters (A-Z, 0-9) and spaces. The comments can be used to describe the script, so so it isn't necessary to rely on the name alone to remember the script's purpose.
2. Click **OK**. The recording process begins. From this point on, until the **Stop Record** button is selected in the Path Analyzer Options dialog box, each command that is executed in the options dialog box will be recorded into a script. This includes the **Run** button for opening the Path Analyzer dialog box, and the **Save** icon for saving paths to a text file. The progress of the script is shown in the **Script Commands** text box; the text can be edited if changes are needed once recording is complete.



To stop the recording process:

1. Click **Stop Record** in the Path Analyzer Options dialog box.

NOTE: The names of the **Stop Record** and **Record** buttons change to indicate if it is currently recording.

23.2.2.1.1.3.2 Save and Execute

Once a script is recorded, click **Save** to save it to the script file (`design_name.qsc`). This is a text file stored in the design directory and automatically loaded whenever the Path Analyzer is started with the appropriate design name.

To reset the Path Analyzer and execute all script commands click **Execute**. A submenu appears. Most commonly used is the **All commands** option. The Update to current line option starts script execution at the beginning of the script and runs until it reaches the selected line in the Script Commands edit box.

All saved scripts for the current design can be listed with the drop-down box (`title: Script`) in the lower left corner of the options dialog box. Once a script is selected, it can be edited using the Script Command text box. To use this dialog box, select a line. Click **X** to delete the line, click the arrows to move the line up or down in the script, and click **new line** to add a new script line above the current one.

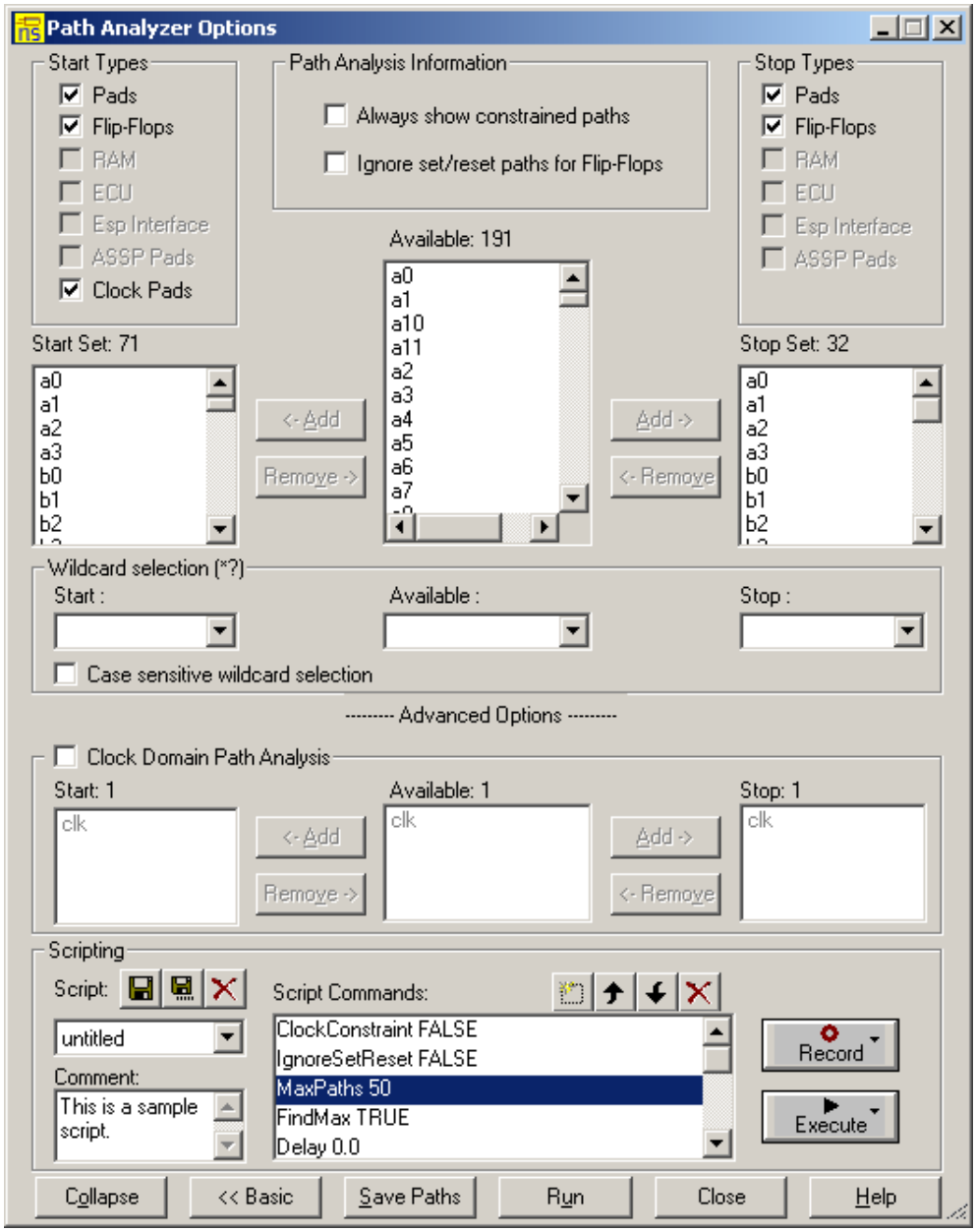
23.2.2.1.1.4 Global Clock Domain Path Analysis

This option is provided to assist the designers to analyze the paths driven by one or more specific global clock(s).

To begin using the Path analyzer's clock domain path analysis function, in the Path Analyzer Options dialog box click **Advanced**. By default, the global clock constraint is turned off. Under advanced options, the boxes in the top frame (labeled Clock Domain Path Analysis) have all the global clocks listed.

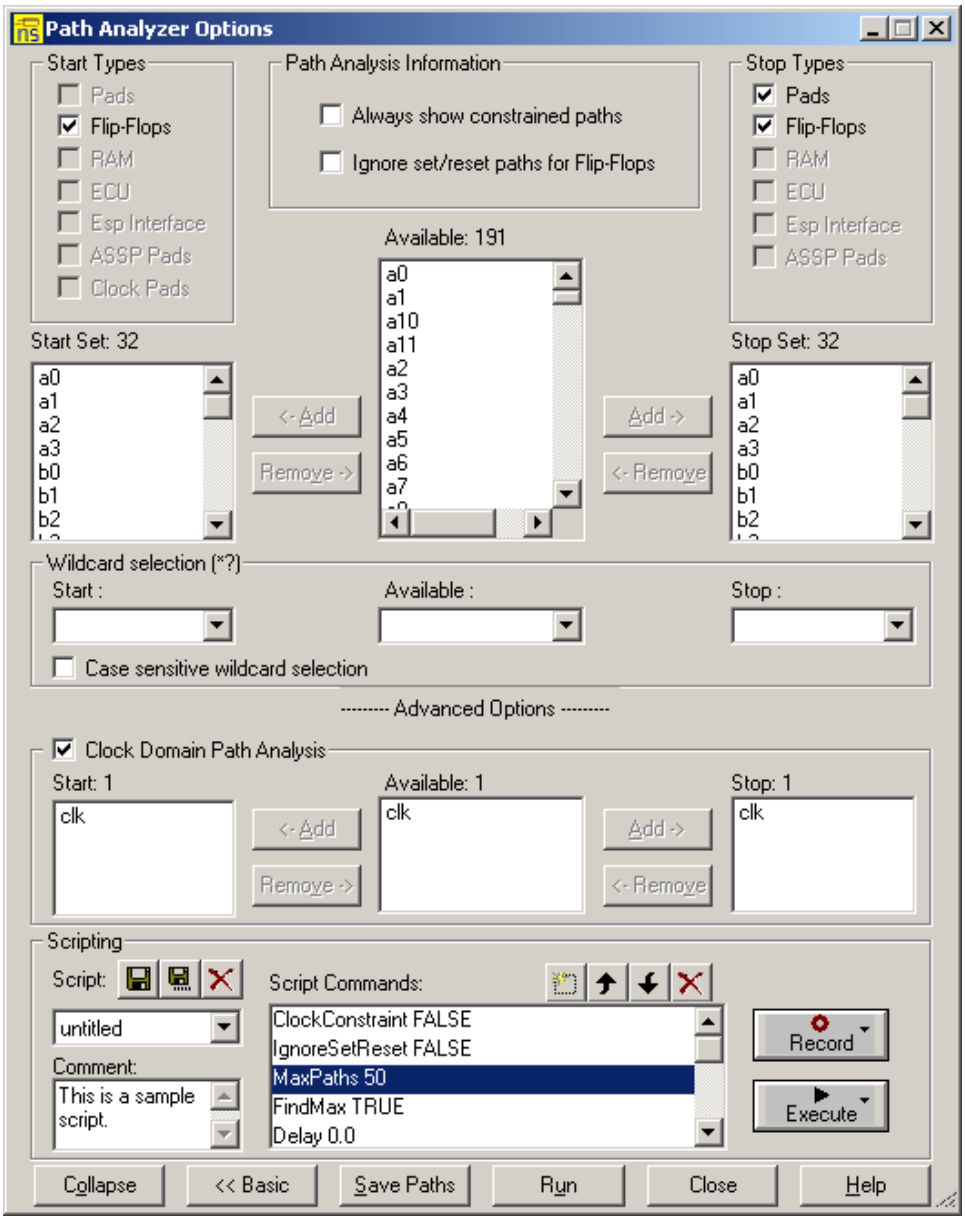
NOTE: If the boxes are grayed out, it means that they are disabled.

© 2008 QuickLogic Corporation



When the global clock domain path analysis is enabled, the Start Set List Box and Stop Set List Box will be refreshed. This is done in such a way that all those nets that are driven by at least one of the global clock net listed in the Start Clock Types List Box or Stop Clock Types

List Box will be enlisted, respectively. Correspondingly, the check boxes in the Start Types and Stop Types frames will also be refreshed. To enable the Clock Domain Path Analysis dialog box, select the **Clock Domain Path Analysis** check box.



The designer can add the global clock nets to the Start, Stop and Available Clock Types Sets to completely mimic the operation between the Start, Stop and Available Sets in the middle frame. The designer can also delete the global clock nets from the Start, Stop and Available Clock Types Sets. Whenever a change is made in the Global Clock Types List Box(es), the Start, Stop, and Available Sets List Boxes are refreshed accordingly.

© 2008 QuickLogic Corporation

23.2.2.1.1.5 Processing False and Multi-Cycle Path in Timing Analyzing

This feature allows users to specify false and multi-cycle path in the design. The Static Timing Analyzer excludes the false paths and adjusts the timing number of the multi-cycle paths in the process of timing analyzing. To use this feature, specify the false and multi-cycle paths in the `.qcf` file in the following format:

```
set_false_path [-from from_list] [-to to_list] [-through through_list]
set_multicycle_path <number of clock cycles> [-from from_list] [-to to_list]
[-through through_list]
```

NOTE: The from-to option provides the ability to specify the start and end nets of a false/multi-cycle path. The through option provides the ability to specify the passing through nets for the false/multi-cycle path.

The designer can use the wildcard in the net names. The wildcards must be a suffix to the net name.

Examples of qcf Commands in the `.qcf` file:

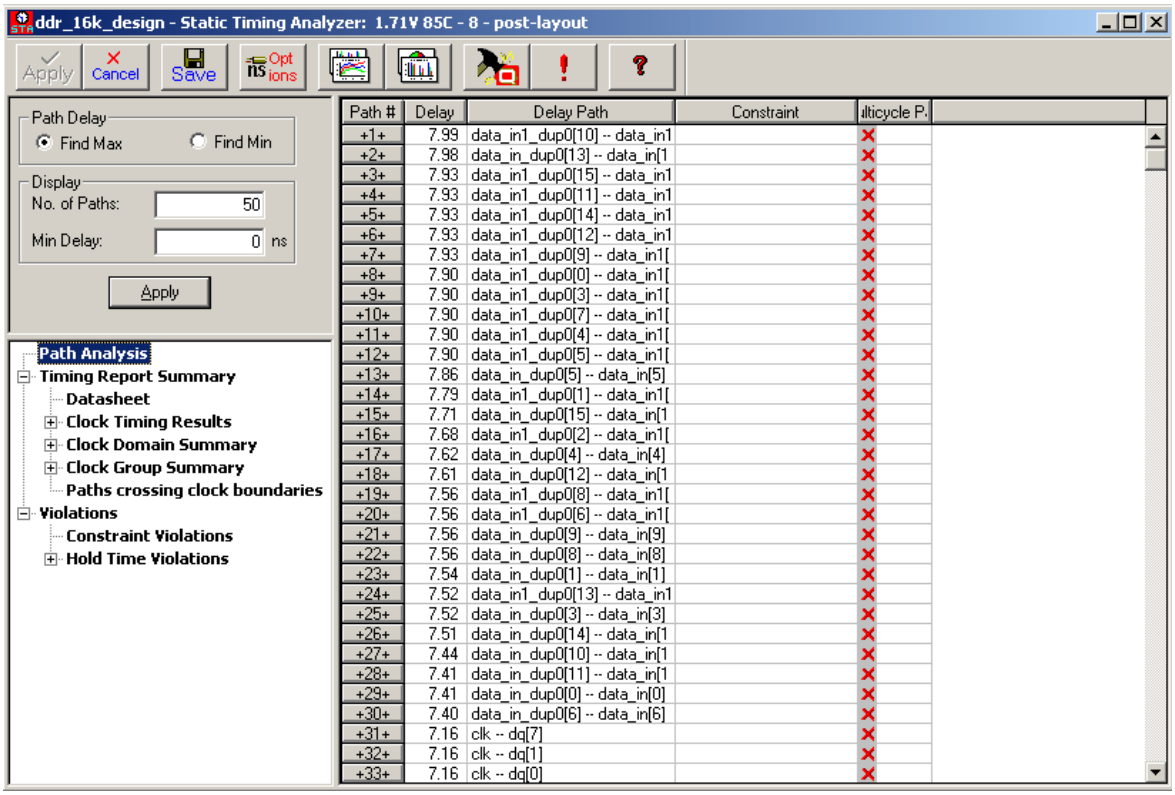
```
set_false_path  -from  net_A  -to  net_B
set_multicycle_path  2  -from  net_C  -to  net_D  -through  net_E
```

The false/multi-cycle path constraints are read into the SpDE internal database as part of the loading process. When the designer runs the SpDE timing analyzing tools, the false/multi-cycle path constraints are processed, and the results are reflected in the Static Timing Analyzer.

© 2008 QuickLogic Corporation

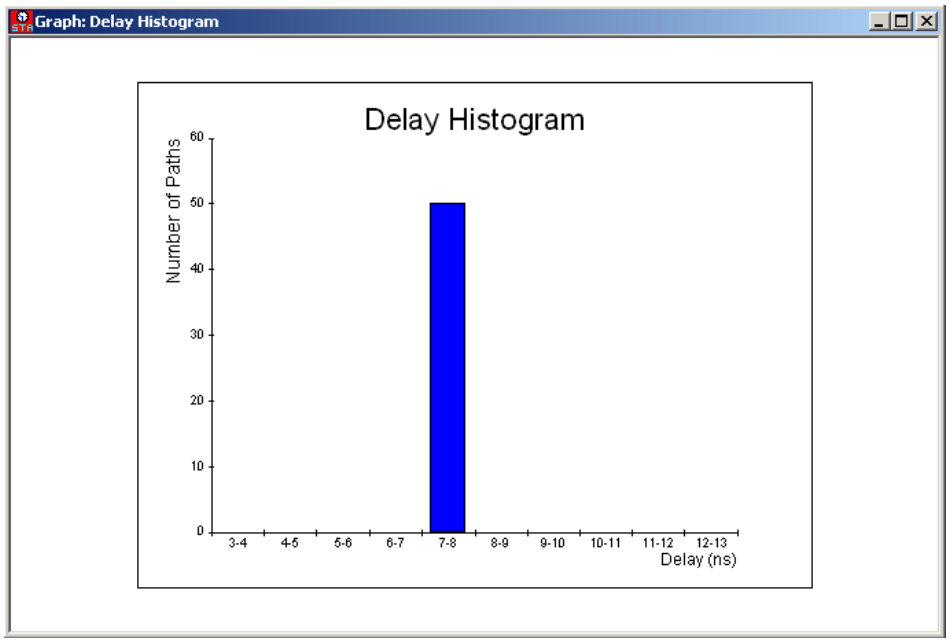
23.2.2.1.1.6 Graphing

The Static Timing Analyzer provides essential information about the performance of the design. It may be useful to view this information in a graphical form. The Static Timing Analyzer can be used to create Path vs. Delay and Delay Histogram graphs for the list of paths currently being displayed. To view the two types of graphs, click the appropriate graph icon from the STA toolbar.

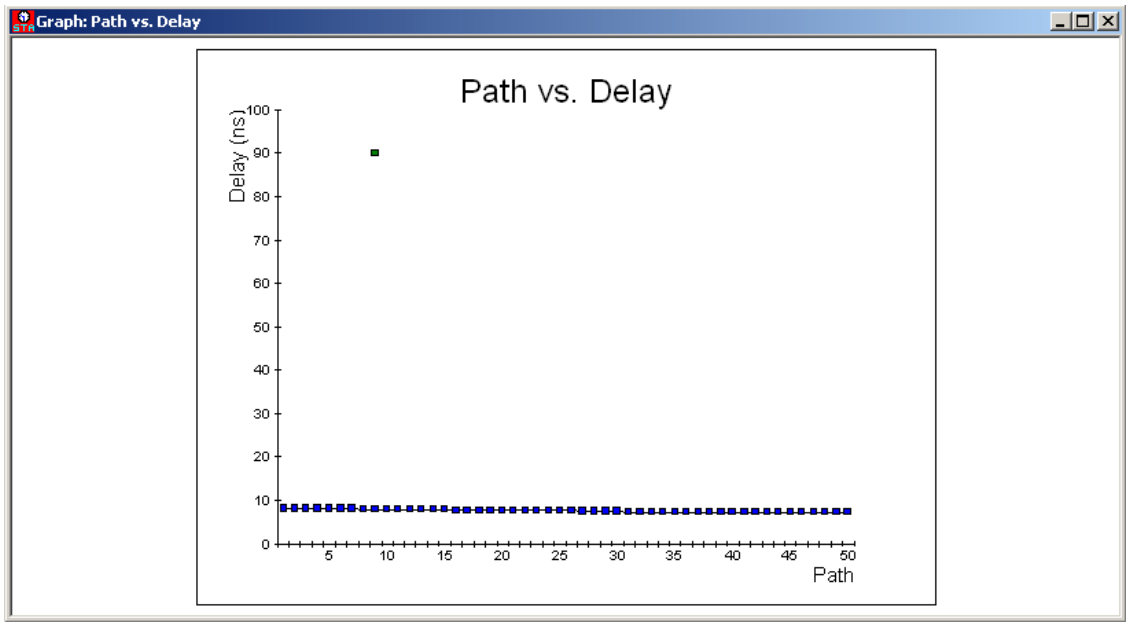

ddr_16k_design - Static Timing Analyzer: 1.71V 85C - 8 - post-layout
Apply
Cancel
Save
Options
Path Delay
Find Max
Find Min
Display
No. of Paths: 50
Min Delay: 0 ns
Apply
Path Analysis
Timing Report Summary
Datasheet
Clock Timing Results
Clock Domain Summary
Clock Group Summary
Paths crossing clock boundaries
Violations
Constraint Violations
Hold Time Violations
Path # | Delay | Delay Path | Constraint | ...lticycle P...
+1+ 7.99 data_in1_dup0[10] -- data_in1
+2+ 7.98 data_in_dup0[13] -- data_in[1
+3+ 7.93 data_in1_dup0[15] -- data_in1
+4+ 7.93 data_in1_dup0[11] -- data_in1
+5+ 7.93 data_in1_dup0[14] -- data_in1
+6+ 7.93 data_in1_dup0[12] -- data_in1
+7+ 7.93 data_in1_dup0[9] -- data_in1[
+8+ 7.90 data_in1_dup0[0] -- data_in1[
+9+ 7.90 data_in1_dup0[3] -- data_in1[
+10+ 7.90 data_in1_dup0[7] -- data_in1[
+11+ 7.90 data_in1_dup0[4] -- data_in1[
+12+ 7.90 data_in1_dup0[5] -- data_in1[
+13+ 7.86 data_in_dup0[5] -- data_in[5]
+14+ 7.79 data_in1_dup0[1] -- data_in1[
+15+ 7.71 data_in_dup0[15] -- data_in[1
+16+ 7.68 data_in1_dup0[2] -- data_in1[
+17+ 7.62 data_in_dup0[4] -- data_in[4]
+18+ 7.61 data_in_dup0[12] -- data_in[1
+19+ 7.56 data_in1_dup0[8] -- data_in1[
+20+ 7.56 data_in1_dup0[6] -- data_in1[
+21+ 7.56 data_in_dup0[9] -- data_in[9]
+22+ 7.56 data_in_dup0[8] -- data_in[8]
+23+ 7.54 data_in_dup0[1] -- data_in[1]
+24+ 7.52 data_in1_dup0[13] -- data_in1
+25+ 7.52 data_in_dup0[3] -- data_in[3]
+26+ 7.51 data_in_dup0[14] -- data_in[1
+27+ 7.44 data_in_dup0[10] -- data_in[1
+28+ 7.41 data_in_dup0[11] -- data_in[1
+29+ 7.41 data_in_dup0[0] -- data_in[0]
+30+ 7.40 data_in_dup0[6] -- data_in[6]
+31+ 7.16 clk -- dq[7]
+32+ 7.16 clk -- dq[1]
+33+ 7.16 clk -- dq[0]

© 2008 QuickLogic Corporation

The Delay Histogram graph uses a range of Path Delays as buckets on the X-axis. The number of paths falling into a delay range bucket is shown as a Y value for each range.



The Path vs. Delay graph shows the Path Delays on the Y-axis and the Path Numbers on the X-axis. Double-clicking on the points in this graph has the same highlighting effect as double-clicking on the paths in Static Timing Analyzer.



23.2.2.1.1.7 Key Calculations

Using the Path Analyzer, key information can be determined with simple arithmetic. Calculations provide a quick and convenient means of determining worst-case design performance. This section describes the Path Analyzer's features for making key calculations.

23.2.2.1.1.7.1 Clock Skew

1. To display the Path Analyzer Options dialog box from STA, select Path Analysis from Selection Pane. It can also be displayed by clicking on the STA toolbar.
2. Set the Path Delay radio button to **Find Max** from Path Analyzer Options pane.
3. In the Start Types check box group, activate **Clock Pad** only.
4. In the Stop Types check box group, activate **Flip-Flops** only.
5. Click **Run** to execute the Path Analyzer.
6. Call the first path listed: `max_clock`.
7. Set the Path Delay radio button to **Find Min** from Path Analyzer Options pane**.**
8. Click **Run** to execute the Path Analyzer.
9. Call the first path listed: `min_clock`.

   Skew = max_clock – min_clock

This clock skew calculation is always pessimistic, since the calculation ignores the fact that clock skew is meaningful only between flip-flops on a common path (i.e., flip-flops with a common clock driver).

23.2.2.1.1.7.2 Operating Frequency

1. To display the Path Analyzer Options dialog box from STA, select Path Analysis from Selection Pane. It can also be displayed by clicking on STA toolbar.
2. Set the Path Delay radio button to **Find Max** from Path Analyzer Options pane.
3. In the Start Types check box group, activate **Flip-Flops** only.
4. In the Stop Types check box group, activate **Flip-Flops** only.
5. Click **Run** to execute the Path Analyzer.
6. Note the delay of the first path listed.

$$F_{max} = \frac{1}{first\_path + SKEW}$$

The operating frequency calculation assumes that the design is fully synchronous with a single clock signal.

Setup Time

1. To display the Path Analyzer Options dialog box from STA, select Path Analysis from Selection Pane. It can also be displayed by clicking on STA toolbar.
2. Set the Path Delay radio button to **Find Max** from Path Analyzer Options pane.
3. In the Start Types check box group, activate **Pads** only.
4. In the Stop Types check box group, activate **Flip-Flops** only. Select only those Flip-Flips that are driven by the corresponding clock.
5. Click **Run** to execute the Path Analyzer.
6. Note the paths and their delays.

© 2008 QuickLogic Corporation

7. Set the Path Delay radio button to **Find Min**.
8. Click **Apply.**
9. In the Start Types check box group, activate **Clock Pad** only.
10. In the Stop Types check box group, activate **Flip-Flops** only. Select only those Flip-Flips that are driven by the corresponding clock.
11. Click **Run** to execute the Path Analyzer.
12. Note the paths and their delays.

    tsetup = Max (pads_to_ffs – clock_to_ffs)

    That is, if there are three Flip-Flops (FF1, FF2, FF3), then setup time is calculated as:

    tsetup = Max ((pads_to_ff1 – clock_to_ff1), (pads_to_ff2 – clock_to_ff2), (pads_to_ff3 – clock_to_ff3))

The Setup Time calculation is always pessimistic, because the two calculations will often apply to different flip-flops.

23.2.2.1.1.7.3 Hold Time

1. To display the Path Analyzer Options dialog box from STA, select Path Analysis from Selection Pane. It can also be displayed by clicking on STA toolbar.
2. Set the Path Delay radio button to **Find Min** from the Path Analyzer Options pane.
3. In the Start Types check box group, activate **Pads** only.
4. In the Stop Types check box group, activate **Flip-Flops** only. Select only those Flip-Flops driven by the corresponding clock.
5. Click **Run** to execute the Path Analyzer.
6. Note the paths and their delays.
7. Set the Path Delay radio button to **Find Max**. Select **Apply**.
8. In the Start Types check box group, activate **Clock Pad** only.
9. In the Stop Types check box group, activate **Flip-Flops** only. Select only those Flip-Flops driven by the corresponding clock.
10. Click **Run** to execute the Path Analyzer.
11. Note the paths and their delays.

23.2.2.1.1.7.4 Clock-to-Output

This clock-to-output calculation helps determine the time from a change on the clock pin to a change on the output pins.

1. To display the Path Analyzer Options dialog box from STA, select Path Analysis from Selection Pane. It can also be displayed by clicking on STA toolbar.
2. Set the Path Delay radio button to **Find Max** from the Path Analyzer Options pane.
3. In the Start Types check box group, activate **Clock Pads** only.
4. In the Stop Types check box group, activate **Flip-Flops** only.

© 2008 QuickLogic Corporation

5. Click the **Run** button to execute the Path Analyzer.
6. Note the delay of the first path listed. Call it: `clock_to_ffs`.
7. Set the Path Delay radio button to **Find Max**.
8. In the Start Types check box group, activate **Flip-Flops** only.
9. In the Stop Types check box group, activate **Pads** only.
10. Click the **Run** button to execute the Path Analyzer.
11. Note the delay of the first path listed. Call it: `ffs_to_pads`.

tclktoq = (clock_to_ffs + ffs_to_pads)

The clock to output calculation will always be pessimistic. The longest clock to flip-flop delay is added to the longest flip-flop to output delay, but often the two longest delays do not refer to the same flip-flop. As a result, this calculation is always equal to or greater than the true clock to output delay.

If a more accurate clock to output calculation is necessary, carefully match up the clock to flip-flop delays to the flip-flop to output delays, so the same flip-flop is always used for both numbers.

NOTE: If a path has a buffer in between its start and end points, and if that is also selected into the start set along with the original start set of path, then the path that starts at the buffer would get reported while the first path would be suppressed.

23.2.2.1.2 ESP Core Clock Timings

If the design utilizes any of the sapphire parts with a Clock signal, the timings such as Period, Setup and ClockToOut of that signal are stored in the corresponding `esp` file. These values are picked up and multiplied by the device k-factor before reporting to the designer under this feature.

23.2.2.2 Timing Report Summary

This section contains the following subsections:

- **"Datasheet" on page 337**
- **"Clock Timing Results" on page 337**
- **"Clock Domain Summary" on page 339**
- **"Clock Group Summary" on page 339**
- **"Paths Crossing Clock Boundaries" on page 340**

Selecting Timing Report Summary disables all of the icons in the STA toolbar except **Save** This also disables the Path Analysis Option Pane.

This section gives the consolidated timing report summary in HTML format. This includes all direct and internally generated clocks, clock domain and clock group clocks.

© 2008 QuickLogic Corporation

Table 23-4 describes the information reported for all clocks.

Table 23-4: Consolidated Timing Report Summary

Item	Description
Clock Name	Name of the clock signal.
Clock Source	Name of the clock source.
Frequency/Period	Clock operating frequency (MHz) or clock period (ns).
Setup Time	The worst setup time (ns).
ClockToOut	The worst clock to out time (ns).
Clock Type	Indicates whether the clock is direct, user-defined, clock domain or a clock group.

23.2.2.2.1 Datasheet

This section gives the details about the I/O pin Information.

Selecting **Datasheet** disables all of the icons in the STA toolbar except **Save**. This also disables the Path Analysis Option Pane.

For all the used I/O pins in the design, the worst setup, clock to out and hold timings are reported under this feature.

The formula used for Setup, ClockToOut and Hold is as follows:

```
Setup Time = Max (PadToFF (max) – ClockToFF (min))
ClockToOut = ClockToFF (max) + FFToPad (max)
Hold Time = Max (ClockToFF (max) – PadToFF (min))
```

The setup and hold times are reported for an Inpad, and ClockToOut is reported for an Outpad. For a Bidir pad, both are reported (see **Table 23-5**).

Table 23-5: I/O Pins

Item	Description
I/O Pad	The name of the pin as shown in the physical viewer.
Signal	The signal connected to the I/O pin.
Type	The pin type (input, output or bidir).
I/O Standard	The standard of the I/O pin (LVTTL, LVCMOS2, etc.).
Usage of FFs	Amount of FFs in the I/O.
Setup Time	Worst setup time if t he pin is an INPUT or BIDIR.
ClkToOut Time	Worst Clock-to-out time if the pin is an OUTPUT or BIDIR.
Hold Time	Worst hold time if the pin is an IPUT or BIDIR.

23.2.2.2.2 Clock Timing Results

Selecting **Clock Timing Results** disables all the icons in the STA toolbar except **Save**. This also disables the Path Analysis Option Pane.

© 2008 QuickLogic Corporation

This section reports the following information in HTML format (see **Table 23-6**).

Table 23-6: Clock Timing Results

Item	Description
Clock Name	Name of the clock signal.
Frequency/Period	Clock operating frequency (MHz) or clock period (ns).
Setup Time	The worst setup time (ns).
ClockToOut	The worst clock to out time (ns).
Clock Type	Indicates whether the clock is direct or user-defined.

Expanding this section displays the list of all direct and constrained clocks.

Expanding each clock displays further subsections such as:

- **"Sync --> Sync" on page 338**
- **"Pad --> Sync" on page 338**
- **"Sync --> Pad" on page 338**
- **"Sync --> Pad" on page 338**

23.2.2.2.2.1 Sync --> Sync

Selecting **Sync-->Sync** disables all the icons in the STA toolbar except **Save**.

This section reports the paths that start sets have FF/RAM/QMATH and stop sets have FF/RAM/QMATH.

The Path Analysis Options pane is enabled, since this lists the paths. Based on the **Find Max** or **Find Min** selection, it displays the paths accordingly. Similarly, typing the number of paths and delay lists only those paths that satisfy the given criteria.

23.2.2.2.2.2 Pad --> Sync

Selecting **Pad --> Sync** disables all the icons in the STA toolbar except **Save**.

This section reports the paths that start sets have PADS and stop sets have FF/RAM/QMATH.

The Path Analysis Options pane is enabled, since this lists the paths. Based on the **Find Max** or **Find Min** selection, it displays the paths accordingly. Similarly, typing the number of paths and delay lists only those paths that satisfy the given criteria.

23.2.2.2.2.3 Sync --> Pad

Selecting **Sync --> Pad** disables all the icons in the STA toolbar except **Save**.

This section reports the paths that start sets have FF/RAM/QMATH and stop sets have PADS.

The Path Analysis Options pane is enabled, since this lists the paths. Based on the **Find Max** or **Find Min** selection, it displays the paths accordingly. Similarly, typing the number of paths and delay lists only those paths that satisfy the given criteria.

© 2008 QuickLogic Corporation

23.2.2.2.2.4 Clock --> Sync

Selecting **Clock --> Sync** item disables all the icons in the STA toolbar except **Save**.

This section reports the paths that start sets have clocks/clock domains/clock groups and stop sets have FF/RAM/QMATH.

The Path Analysis Options pane is enabled, since this lists the paths. Based on the **Find Max** or **Find Min** selection, it displays the paths accordingly. Similarly, typing the number of paths and delay lists only those paths that satisfy the given criteria.

23.2.2.2.3 Clock Domain Summary

Selecting **Clock Domain Summary** disables all the icons in the STA toolbar except **Save**. This also disables the Path Analysis Option Pane.

As shown in **Table 23-7**, Clock Domain Summary reports information for each clock source in HTML format.

Table 23-7: Clock Domain Summary

Item	Description
Clock Source	Name of the clock source.
Frequency/Period	Clock operating frequency (MHz) or clock period (ns).
Setup Time	The worst setup time (ns).
ClockToOut	The worst clock to out time (ns).
Clock Skew	The worst case skew of the clock signal.

On expansion of this section, it will display the list of all clock domains.

On expansion of each clock domain, it has further sub-sections like:

- Sync --> Sync (see **"Sync --> Sync" on page 338**)
- Pad --> Sync (see **"Pad --> Sync" on page 338**)
- Sync --> Pad (see **"Sync --> Pad" on page 338**)
- Clock --> Sync (see **"Sync --> Pad" on page 338**)

23.2.2.2.4 Clock Group Summary

On selection of Clock Group Summary, all the icons in the STA toolbar except **Save** are disabled. This also disables the Path Analysis Option Pane.

As shown in **Table 23-8**, the Clock Group Summary reports information for each clock group in HTML format.

Table 23-8: Clock Group Summary

Item	Description
Clock Name	Name of the clock signal.
Clock Source	Name of the clock source.
Frequency/Period	Clock operating frequency (MHz) or clock period (ns).
Setup Time	The worst setup time (ns).

© 2008 QuickLogic Corporation

Table 23-8: Clock Group Summary (Continued)

Item	Description
ClockToOut	The worst clock to out time (ns).
Clock Type	Indicates whether the clock is direct, user-defined, clock domain or a clock group.
Phase	The phase difference.
Jitter	

Expanding this section displays the list of all clock groups.

Expanding each clock group displays further subsections such as:

- Sync --> Sync (see **"Sync --> Sync" on page 338**)
- Pad --> Sync (see **"Pad --> Sync" on page 338**)
- Sync --> Pad (see **"Sync --> Pad" on page 338**)
- Clock --> Sync (see **"Sync --> Pad" on page 338**)

23.2.2.2.5 Paths Crossing Clock Boundaries

Selecting **Paths Crossing Clock Boundaries** disables all the icons in the STA toolbar except **Save**. In the Path Analysis Option Pane, only the **No. of paths** is enabled.

These are the paths that start in one clock domain and end in another clock domain. As shown in **Table 23-9**, the Paths Crossing Clock Boundaries section reports the paths that start FF/RAM/QMATH and stop FF/RAM/QMATH and are clocked by different clocks.

Table 23-9: Paths Crossing Clock Boundaries

Item	Description
Path No.	The serial number of the path.
Source Clock	Starting clock domain.
Destination Clock	Ending clock domain.
Delay	The total delay of the path.
Delay Path	The complete subpath information which includes the trails or routes along with the delay.

23.2.2.3 Violations

Selecting **Violations** from the tree disables all the icons in the STA toolbar except **Save**. This also disables the Path Analysis Option Pane.

This section gives details about any violations in the design and contains the following subsections:

- **"Constraint Violations" on page 341**
- **"Hold Time Violations" on page 341**
  - **"Clock (Direct and Constrained)" on page 341**
  - **"Clock Domain" on page 342**
  - **"Clock Group" on page 343**

© 2008 QuickLogic Corporation

23.2.2.3.1 Constraint Violations

Selecting **Constraint Violations** from the tree disables all the icons in the STA toolbar except **Save**. This also disables the Path Analysis Option Pane.

As shown in **Table 23-10**, the Constraint Violations section shows the details for each constraint specified by the user.

Table 23-10: Constraints

Item	Description
Path Name	Constraint object such as clock, false path, multicycle path, etc.
Type	The name of the constraint parameter.
Constraint	Constraint value given for the parameter.
Miss	Reports whether the constraint is missed or not (** or nothing).
Result	Results from the tool for the given constraint parameter.
Slack	Slack for the parameter.

If there are any false and multicycle path constraints, in addition to the above report, those are also reported in the following sections.

False Path Constraints—The designer provides the start and stop nets of the paths to be considered as false paths. The reporting format of such paths is as follows:

```
1 <start netname> to <end netname>
2 <start netname> to <end netname>
```

Multicycle Path Constraints—The designer provides the start and stop nets of the paths, along with the number of clock period cycles, to be considered as multicycle paths. The reporting format of such paths is as follows:

```
1 <start netname> to <end netname>, <number of clock period cycles>
2 <start netname> to <end netname>, <number of clock period cycles>
```

If there are no constraints found in the QCF file, it displays "`No constraints found in QCF file.`"

23.2.2.3.2 Hold Time Violations

Selecting **Hold Time Violations** disables all the icons in the STA toolbar except **Save**. This also disables the Path Analysis Option Pane.

This section gives details about the shortest paths for clocks, clock domains or clock groups and has further subsections such as:

- Clock (Direct and Constrained)
- Clock Domain
- Clock Group

23.2.2.3.2.1 Clock (Direct and Constrained)

Selecting **Clock (Direct and Constrained)** disables all the icons in the STA toolbar except **Save**. This also disables the Path Analysis Option Pane.

© 2008 QuickLogic Corporation

As shown in **Table 23-11**, this section gives details about the shortest paths for each direct and internally generated clock.

The formula for the short path delay is:

```
ShortPathDelay = FFmToFFn (Min) + ClockToFFm (Min) – ClockToFFn (Max) - Hold
Time of FFn
```

In the above equation, both m and n flip-flops (m # n) are driven by the same Clock signal. The first term *FFmToFFn Min Delay* represents the path delay between the m and n flip-flops for the best case (Minimum delay). The second minus the third terms gives the clock skew for FFm and FFn destinations. After subtracting the skew from the Flop-to-Flop delay, the Hold time of FFn is again subtracted. If the resulting Short Path Delay is less than 0.5 ns, but above or equal to zero, then those paths are close to hold time violations. Alternatively, if the Short Path Delay is negative, those paths have already been violating the hold time at FFn. Both of the above paths are reported to the designer.

Table 23-11: Shortest Paths for Clocks

Item	Description
Path Name	The serial number of the path.
Delay	The total delay of the path.
Short Path Delay	The result of formula (1).
Delay Path	The complete subpath information which includes the trails or routs along with the delay.

23.2.2.3.2.2 Clock Domain

Selecting **Clock Domain** disables all the icons in the STA toolbar except **Save**. This also disables the Path Analysis Option Pane.

As shown in **Table 23-12**, this section gives details about the shortest paths for each clock domain.

The formula for the short path delay is:

```
ShortPathDelay = FFmToFFn (Min) + ClockToFFm (Min) – ClockToFFn (Max) - Hold
Time of FFn
```

In the above equation, both m and n flip-flops (m # n) are driven by the same Clock signal. The first term *FFmToFFn Min Delay* represents the path delay between the m and n flip-flops for the best case (Minimum delay). The second minus the third terms gives the clock skew for FFm and FFn destinations. After subtracting the skew from the Flop-to-Flop delay, the Hold time of FFn is again subtracted. If the resulting Short Path Delay is less than 0.5 ns, but above or equal

© 2008 QuickLogic Corporation

to zero, then those paths are close to hold time violations. Alternately, if the Short Path Delay is negative, those paths have been already violating the hold time at FFn. Both of the above paths is reported to the designer.

Table 23-12: Shortest Paths for Clock Domain

Item	Description
Path Name	The serial number of the path.
Delay	The total delay of the path.
Short Path Delay	The result of formula (1).
Delay Path	The complete subpath information which includes the trails or routs along with the delay.

23.2.2.3.2.3 Clock Group

Selecting **Clock Group** disables all the icons in the STA toolbar except **Save**. This also disables the Path Analysis Option Pane.

As shown in **Table 23-13**, this section gives details about the shortest paths for each clock group.

The formula for the short path delay is:

```
ShortPathDelay = FFmToFFn (Min) + ClockToFFm (Min) – ClockToFFn (Max) - Hold
Time of FFn
```

In the above equation, both m and n flip-flops (m # n) are driven by the same Clock signal. The first term *FFmToFFn Min Delay* represents the path delay between the m and n flip-flops for the best case (Minimum delay). The second minus the third terms gives the clock skew for FFm and FFn destinations. After subtracting the skew from the Flop-to-Flop delay, the Hold time of FFn is again subtracted. If the resulting Short Path Delay is less than 0.5 ns, but above or equal to zero, then those paths are close to hold time violations. Alternately, if the Short Path Delay is negative, those paths have been already violating the hold time at FFn. Both of the above paths is reported to the designer.

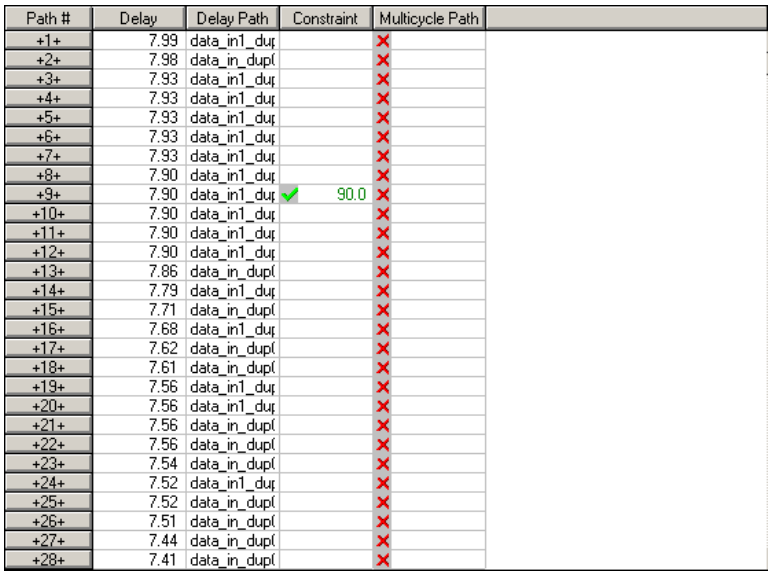
Table 23-13: Shortest Paths for Clock Group

Item	Description
Path Name	The serial number of the path.
Delay	The total delay of the path.
Short Path Delay	The result of formula (1).
Delay Path	The complete subpath information which includes the trails or routs along with the delay.

© 2008 QuickLogic Corporation

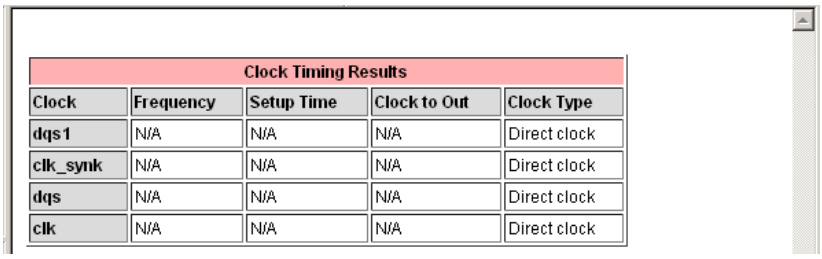
23.2.3 Viewer Pane

The Viewer Pane displays the paths specified in the Options dialog box. It displays the information depending on the selection of tree items from the Selection Pane.



Path #	Delay	Delay Path	Constraint	Multicycle Path
+1+	7.99	data_in1_dup		
+2+	7.98	data_in_dupl		
+3+	7.93	data_in1_dup		
+4+	7.93	data_in1_dup		
+5+	7.93	data_in1_dup		
+6+	7.93	data_in1_dup		
+7+	7.93	data_in1_dup		
+8+	7.90	data_in1_dup		
+9+	7.90	data_in1_dup	90.0	
+10+	7.90	data_in1_dup		
+11+	7.90	data_in1_dup		
+12+	7.90	data_in1_dup		
+13+	7.86	data_in_dupl		
+14+	7.79	data_in1_dup		
+15+	7.71	data_in_dupl		
+16+	7.68	data_in1_dup		
+17+	7.62	data_in_dupl		
+18+	7.61	data_in_dupl		
+19+	7.56	data_in1_dup		
+20+	7.56	data_in1_dup		
+21+	7.56	data_in_dupl		
+22+	7.56	data_in_dupl		
+23+	7.54	data_in_dupl		
+24+	7.52	data_in1_dup		
+25+	7.52	data_in_dupl		
+26+	7.51	data_in_dupl		
+27+	7.44	data_in_dupl		
+28+	7.41	data_in_dupl		

The Viewer Pane displays the timing information and violations in HTML format.



Clock Timing Results

Clock	Frequency	Setup Time	Clock to Out	Clock Type
dqs1	N/A	N/A	N/A	Direct clock
clk_synk	N/A	N/A	N/A	Direct clock
dqs	N/A	N/A	N/A	Direct clock
clk	N/A	N/A	N/A	Direct clock

As shown in **Table 23-14**, the selected path information is displayed in a five-column spreadsheet format.

Table 23-14: Path Information

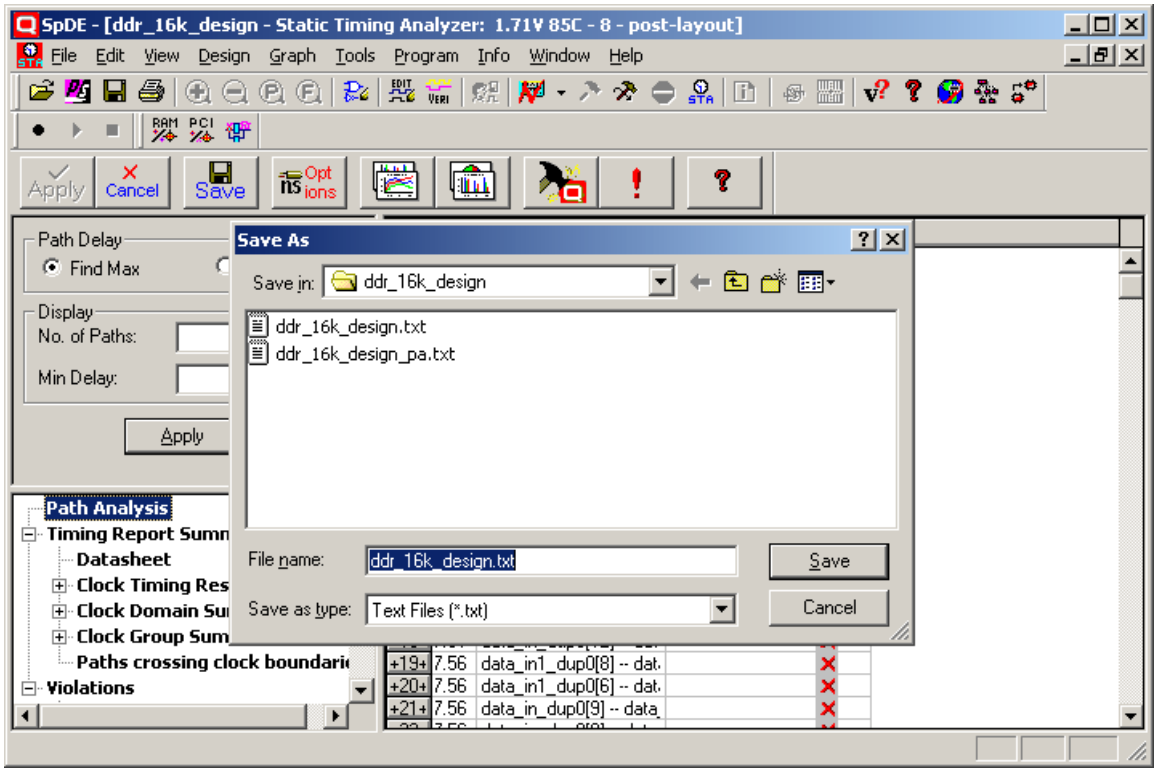
Item	Description
Path No.	Displayed in a push-button format. Double-click the button to expand the path.
Delay	Indicates the respective delay in nanoseconds.
Delay Path	Displays the starting and ending nets of each path.
Constraint	Indicates the desired delay for the delay path.
Multicycle Path	Indicates whether a particular path is a multicycle path or inconsistent with the number of cycles associated with it.

For more information on HTML format, refer to **"Timing Report Summary" on page 336** and **"Violations" on page 340**.

© 2008 QuickLogic Corporation

23.2.3.1 Saving the List of Generated Paths

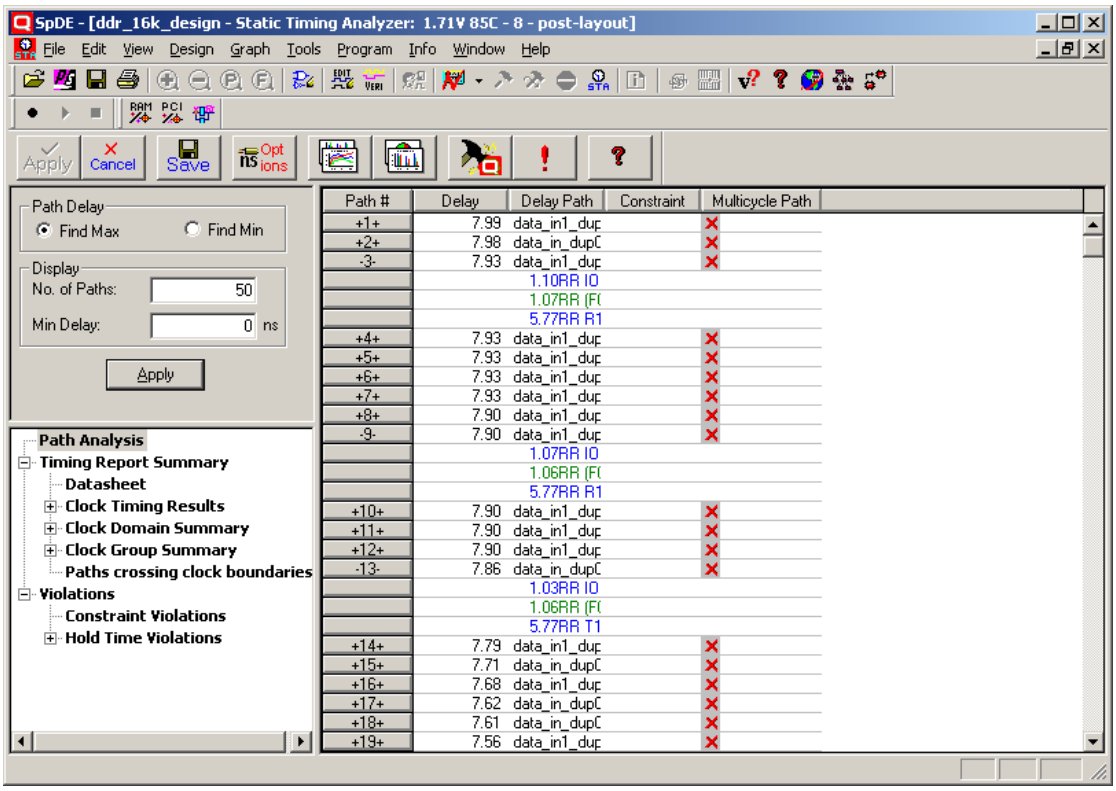
To save the listed paths in a text file, use the Static Timing Analyzer toolbar button. This saves the paths displayed in a Static Timing Analyzer Viewer pane. Select **File>Save** to save the detailed paths in a text file. In both the cases the designer is prompted for the file name.



23.2.3.2 Expanding Paths

To expand a path to see its component sub-paths, position the cursor over the desired button in the Path # column and double-click. As shown below, the Path # button changes from +2+ to -2- to indicate that the path has been expanded. The sub-paths are indented and listed in different colors (blue for sub-path trail and green for sub-path route) respectively, to differentiate them from the other Delay Path rows. Each sub-net lists it's respective delay in

nanoseconds with either R or F token to denote a rising-edge or a falling-edge delay, respectively. Total number of fan-outs of any sub-net is shown in the parentheses. (FO = <fan-out>).



© 2008 QuickLogic Corporation

23.2.4 Static Timing Analyzer Toolbar

As shown in **Table 23-15**, the Static Timing Analyzer toolbar has 10 buttons: **Apply**, **Cancel**, **Save**, **Options**, **Path Vs Delay**, **Delay Histogram**, **Run Tools**, **Path Constraints**, **Priority Router**, and **Help**.

Apply, **Options**, **Path Vs Delay**, **Delay Histogram**, **Run Tools**, **Priority Router** buttons are displayed only for the selection of Path Analysis tree items from the Selection pane.

Table 23-15: SpDE Menu Icons

Icon	Button Command	Function
	Apply	Applies the Constraints and Routing priority
	Cancel	Closes the Static Timing Analyzer.
	Save	Saves the information displayed in a viewer pane to a text file, and prompts for the file name.
	Options	Changes the Path Analyzer options and runs the Path Analyzer.
	Path Vs Delay	Displays Path Vs Delay graph.
	Delay Histogram	Displays Delay Histogram graph.
	Run Tools	Runs the tools after entering constraints.
	Routing Priority	Toggles between Path Analyzer and Priority Router mode.
	Help	Opens Static Timing Analyzer Help.

© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation

Chapter 24
OrCAD Schematic

This chapter describes OrCAD Project Creation and contains the following sections:

- **"Introduction" on page 349**
- **"OrCAD Project Creation" on page 349**
- **"Creating OLB Files in OrCAD" on page 350**

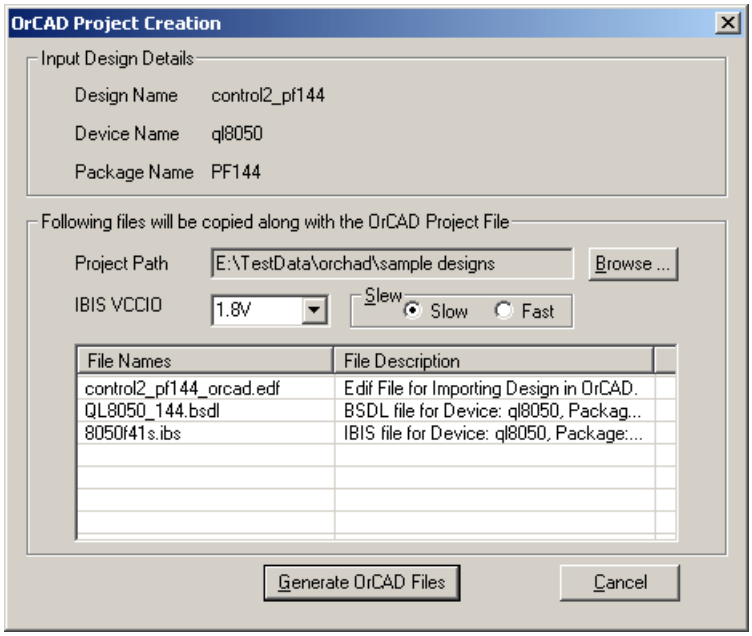
24.1 Introduction

This tool helps in reducing the time and potential user-error to enter pin and property information for PCB schematic creation for large pin-count devices. It also provides BSDL, IBIS and package view (`.bmp`) files along with the software.

24.2 OrCAD Project Creation

This interface is available in the package view of SpDE.

1. Open a `.qdf` or `.edf` file in SpDE.
2. Run the logic optimizer and placer tools.
3. Select **Design>OrCAD Schematic** from the SpDE menu bar. The OrCAD Project Creation box appears.



4. Select the project path and other options. The list of files copied is displayed.

© 2008 QuickLogic Corporation

5. Click on **Generate OrCAD Files**. The files are generated in the project directory.
6. Select **View>Package View** from the SpDE menu bar. The package view appears.
7. Select **View>Generate Package Bitmap** from the SpDE menu bar. The package view is saved as a bitmap file.

NOTE: At present, BSDL and IBIS files are provided only for the following parts: QL8050, QL8150, QL8250, and QL8325.

24.3 Creating OLB Files in OrCAD

To create an `.olb` file

1. Open **Orcad Capture**.
2. Create a new Schematic project.
3. From **File** menu, click on **Import Design** tab.
4. Choose Edif option in the dialog.
5. Browse to find the `.edf` file generated from SpDE.
6. Create a blank `.cfg` file.
7. Click **OK**.
8. An `.olb` file is created. (Check the Session Log to view the name of file created.)
9. Add the `.olb` file. A part Package is created.

The designer can use the generated part symbols in PCB design.

© 2008 QuickLogic Corporation

Chapter 25
SpDE Menu Command Reference

This chapter provides an overview of the SpDE tool and its menu usage. It contains the following sections:

- **“What is SpDE?” on page 351**
- **“Viewer” on page 355**
- **“Design Verifier” on page 366**
- **“Transcript Window” on page 367**
- **“File Menu” on page 370**
- **“View Menu” on page 371**
- **“Design Menu” on page 377**
- **“Tools Menu” on page 378**
- **“Program Menu” on page 381**
- **“Info Menu” on page 382**

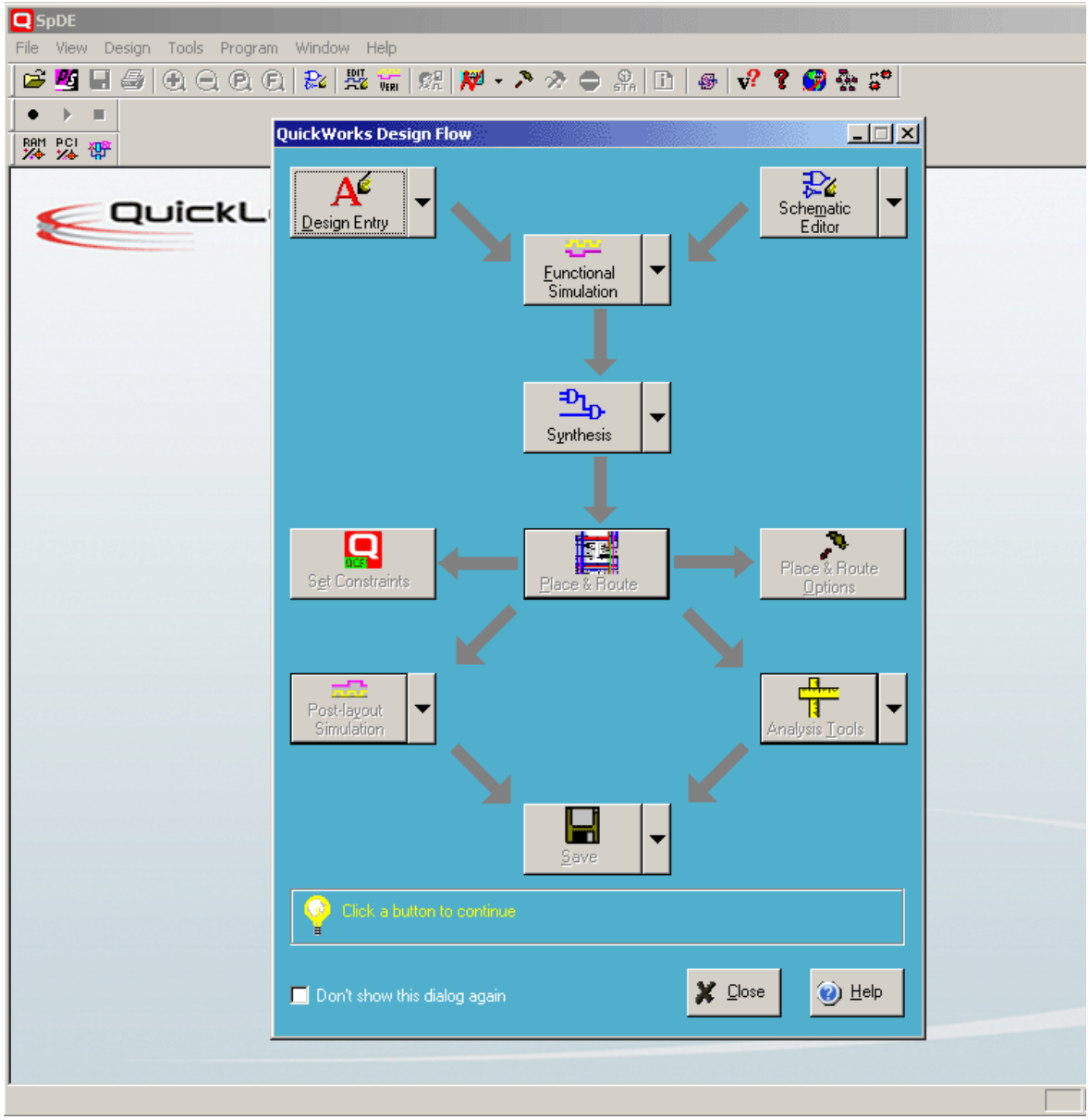
25.1 What is SpDE?

SpDE (Seamless pASIC Design Environment) is a set of quality logic optimization, placement and routing, delay modeling, design analysis, and programming tools—all combined with an easy-to-use graphical interface.

is the desktop icon for SpDE. To start SpDE, double-click on the **SpDE** icon or select it through Windows’ **Start>Programs** menu.

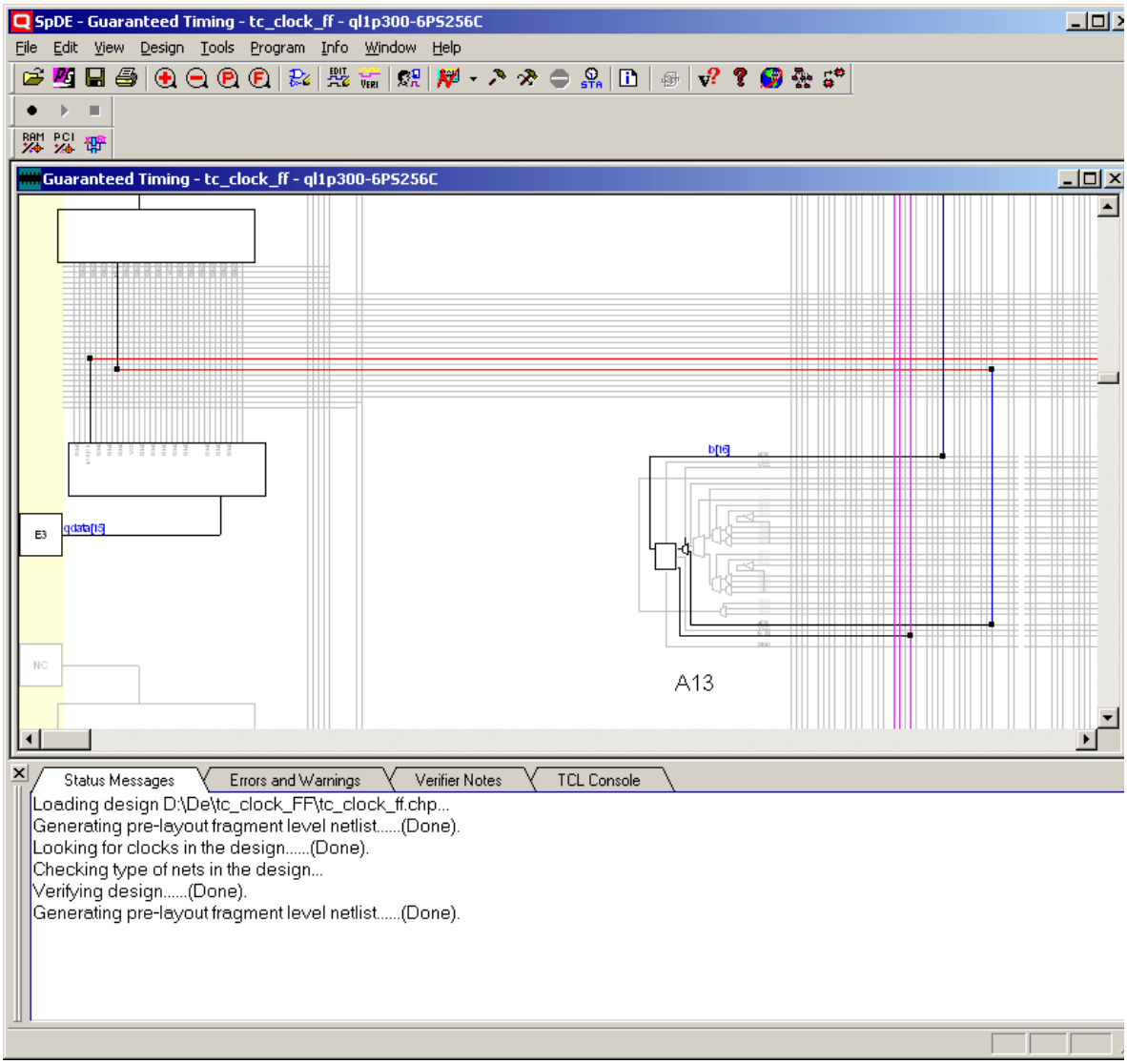
© 2008 QuickLogic Corporation

Once SpDE is launched, the main SpDE dialog box appears along with the QuickWorks Design Flow dialog box. For quick access, the QuickWorks Design Flow dialog box contains the main operations of SpDE in one area. Many of the options normally available from the menus and the tool bar are grayed-out or unavailable initially.



© 2008 QuickLogic Corporation

When a QDIF (.QDF), EDIF (.ED*) or Chip (.CHP) file is loaded into SpDE, a fully functional SpDE dialog box opens.


SpDE - Guaranteed Timing - tc_clock_ff - ql1p300-6P5256C
File Edit View Design Tools Program Info Window Help
Guaranteed Timing - tc_clock_ff - ql1p300-6P5256C
A13
Status Messages
Errors and Warnings
Verifier Notes
TCL Console
Loading design D:\De\tc_clock_FF\tc_clock_ff.chp...
Generating pre-layout fragment level netlist.....(Done).
Looking for clocks in the design.....(Done).
Checking type of nets in the design...
Verifying design.....(Done).
Generating pre-layout fragment level netlist.....(Done).

The title bar at the top of the SpDE dialog box contains both the design name and the full device number for the QuickLogic chip used.

The transcript window at the bottom shows the status messages, errors and warnings, as well as verifier notes. For more details on the transcript window, refer to **"Transcript Window" on page 367**.

The status bar at the very bottom of the SpDE dialog box displays messages that indicate the progress or status for the various SpDE tools.

Use the standard scroll bars to move the viewing area of the dialog box around to different portions of the chip.

The menu bar allows access to all the commands available in SpDE.

NOTE: For a description of SpDE icons, see **Appendix F, "QuickWorks Icons" on page 551**.

© 2008 QuickLogic Corporation

The Physical Viewer dialog box in SpDE displays the result of placement and routing. Blue and black wires are segmented routing. Magenta wires (stemming from the center of the device) represent the clock networks. Red wires are quad wires. Green wires are express wires. For more details see **“Router” on page 155**.

© 2008 QuickLogic Corporation

25.2 Viewer

When a net-list (`qdf, edf, chp`) is opened in SpDE, it is shown graphically in the main SpDE dialog box. The design can be viewed in the following two types of viewers:

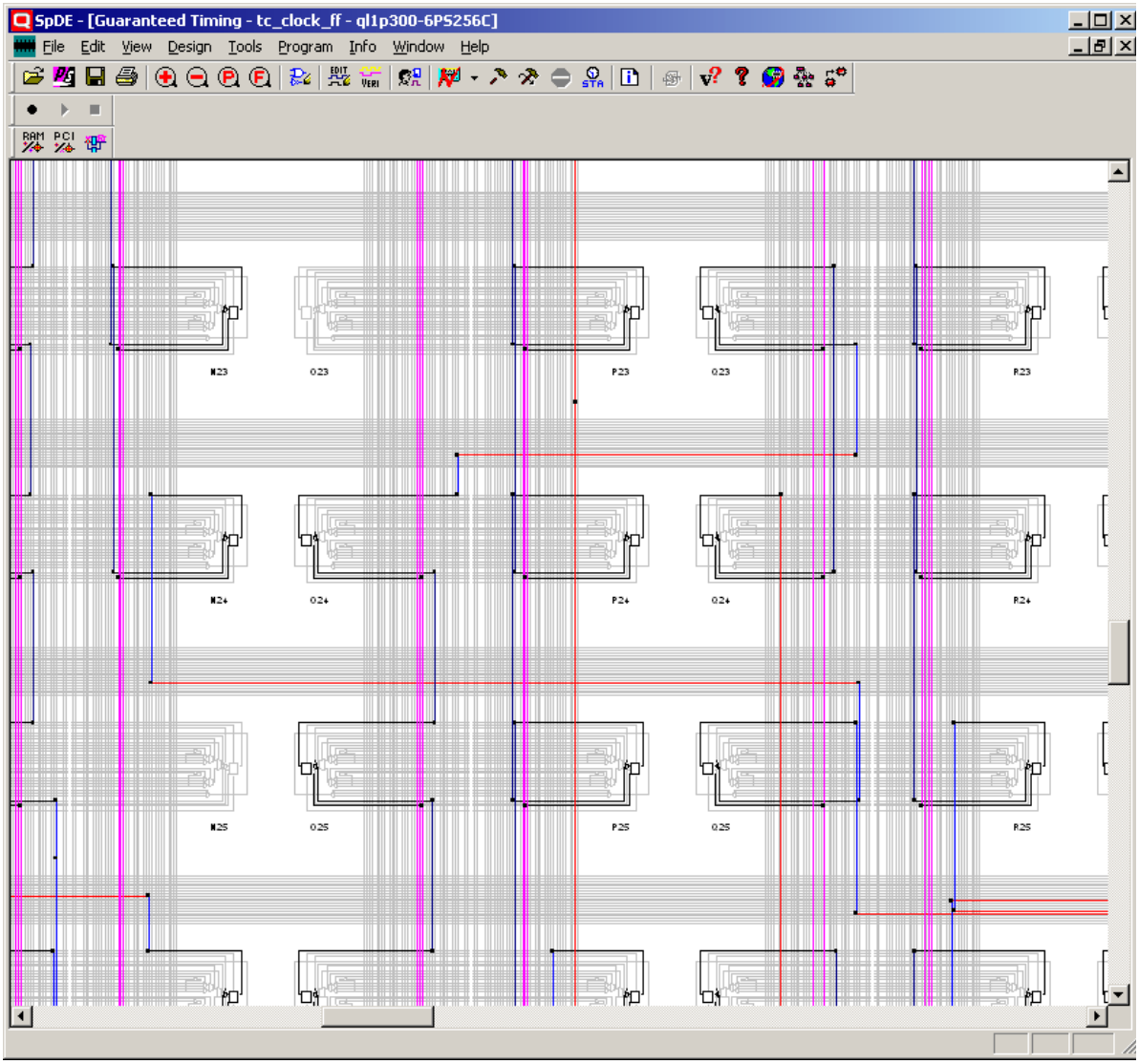
- Physical Viewer
- Package Viewer

25.2.1 Physical Viewer

The physical viewer displays a simplified view of the logic resources, including RAM and ECU blocks, along with the routing channels. The IOs are shown at the peripheral. When a design is opened, it is shown in the physical viewer by default. When a `qdf/edf` is loaded for the first time, all of the resources are shown in gray color. After the tools are run (Place & Route) the viewer shows the placement of resources and their routing.

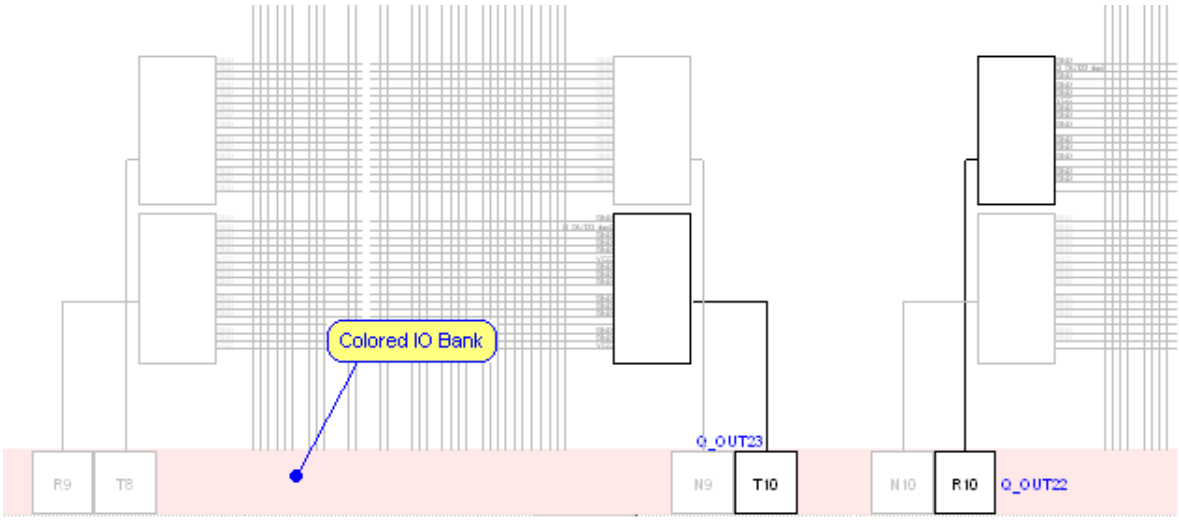
© 2008 QuickLogic Corporation

The following is a snapshot view after Place & Route. The resources used are shown in different colors, other than gray. The user can customize the viewer through the **View>Preferences** menu, including selecting the resources to be shown and choosing resource colors and the text.

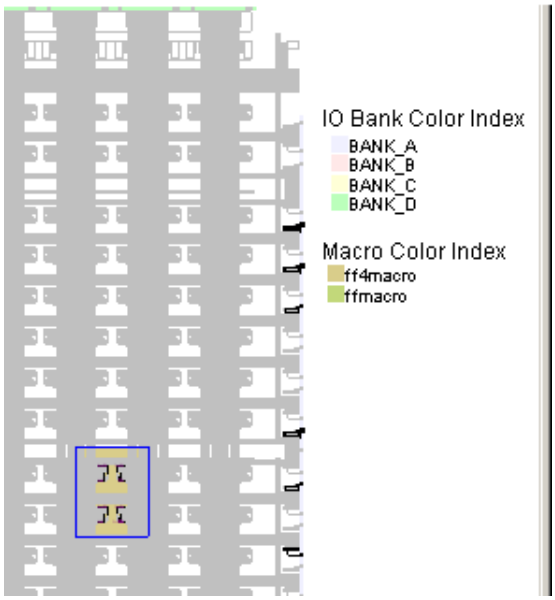


© 2008 QuickLogic Corporation

IO banks in the design are highlighted with different colors.



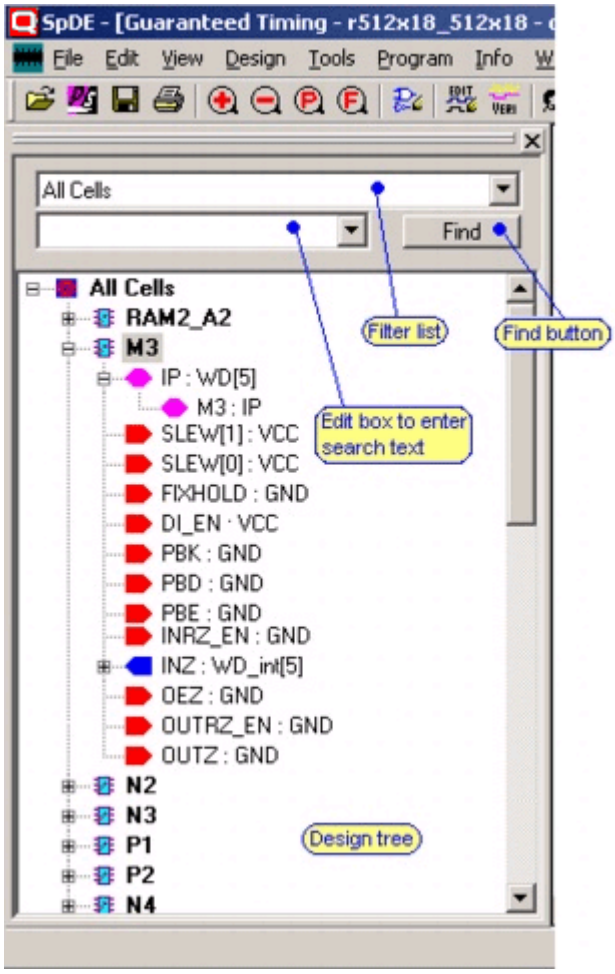
The IO Bank Color Index is shown on the right side of the view.



NOTE: The IO Bank Color Index is visible only in the full fit mode.

25.2.2 Design Browser

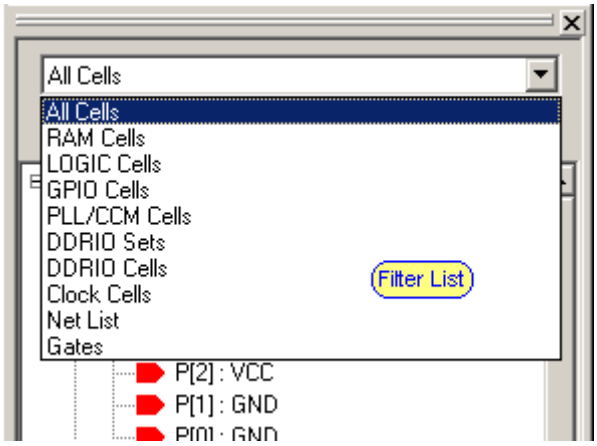
The design browser allows the user to browse design objects and see how they are placed in the physical viewer. To use the design browser select **View>Design Browser**.



© 2008 QuickLogic Corporation

25.2.2.1 Filtering the Design Tree

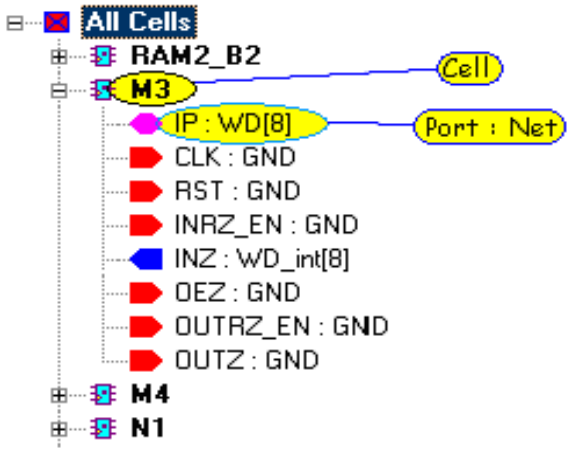
The filter list allows the user to filter the contents of design tree by selecting a filter type from the pull-down menu.



25.2.2.2 Design Tree Structure

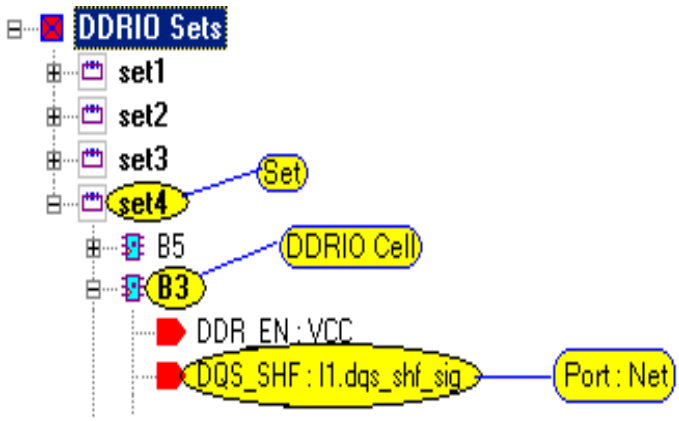
The structure of the design tree depends on the filter type selected as shown in the following examples:

Cell

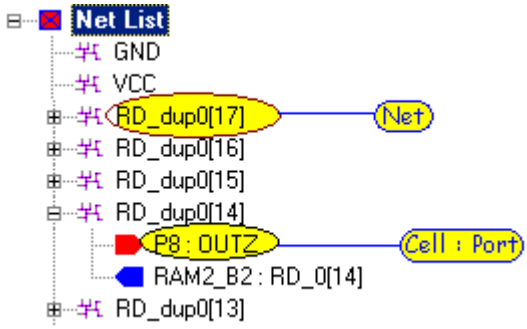


© 2008 QuickLogic Corporation

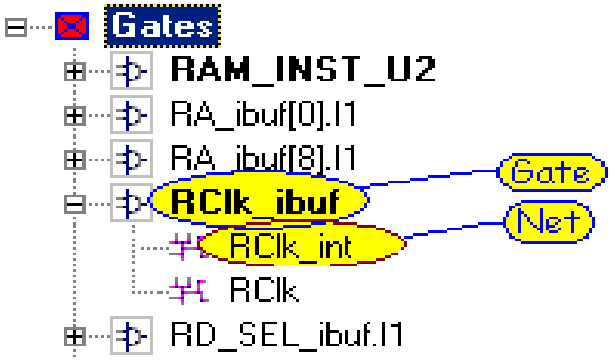
Set



Net



Gate



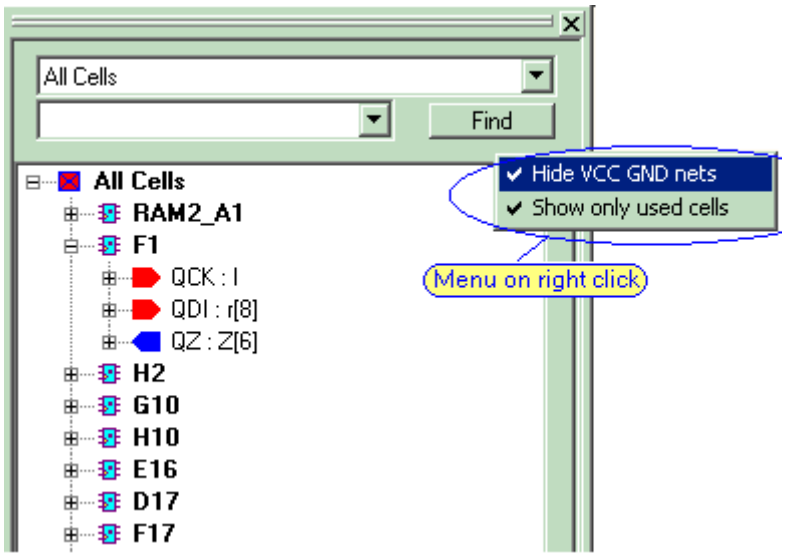
25.2.2.3 Searching in the Design Tree

The design browser allows the user to search for specific text in the design tree. To do this type the text in the edit box and press the **Enter** key or click the **Find** button. The instances of the text searched for are displayed in the status message tab of transcript window. The item in the design tree that has the searched for text is highlighted. Subsequently, pressing the **Enter** key or clicking the **Find** button locates the next item in the design tree that has the searched for text.

© 2008 QuickLogic Corporation

25.2.2.4 Filtering VCC/GND Nets

Right-clicking on the design browser displays a filter menu. Selecting **Hide (or Show) VCC/GND nets** hides (or shows) the VCC/GND nets and the ports with VCC/GND net connections.



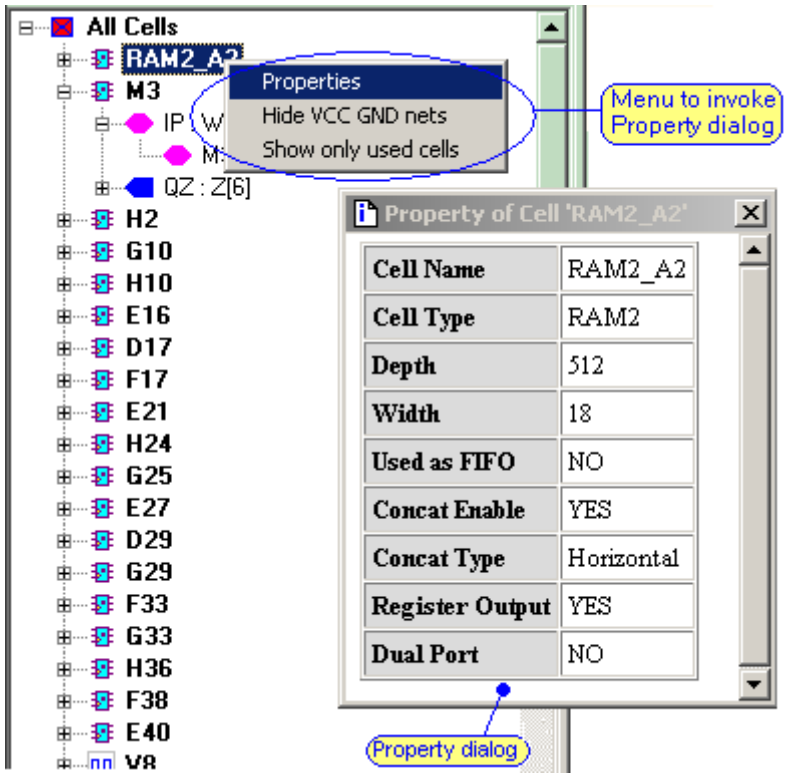
25.2.2.5 Filtering Unused Cells

Right-clicking on the design browser displays a filter menu. Selecting **Hide (or Show) only used cells** hides (or shows) the unused cells in the design.

© 2008 QuickLogic Corporation

25.2.2.6 Object Properties

Right-clicking on an item in the design tree displays a menu that shows the properties of the selected item.



25.2.2.6.1 Property Structures

The property structure varies depending on the type of design object that is selected in the design tree as shown in the following examples:

RAM Cell

Cell Name	RAM2_A2
Cell Type	RAM2
Depth	512
Width	18
Used as FIFO	NO
Concat Enable	YES
Concat Type	Horizontal
Register Output	YES
Dual Port	NO
Read Mode	N/A

© 2008 QuickLogic Corporation

PLL/CCM Cell

Cell Name	CCM_TR
CCM Mode	X2
Phase Shift	90 Deg
Programmable Delay	0.25 ns
Pll Operation	N/A
Pll Mode	N/A
Multiplier	N/A
Divider	N/A

DDRIO/GPIO Cell

Pin Name	M3
Cell Type	GPIO
Bank Name	I/O{B}
IO Standard	DEFAULT

Routable Ports

Port Name	Net Connected	Description
DI_EN	VCC	I/O signal enable controlling DIEB of the GPIO driver.
CLK	GND	Clock signal for optional registers inside PREIOs.
RST	GND	Reset for optional registers inside PREIOs.

Static Logic Ports

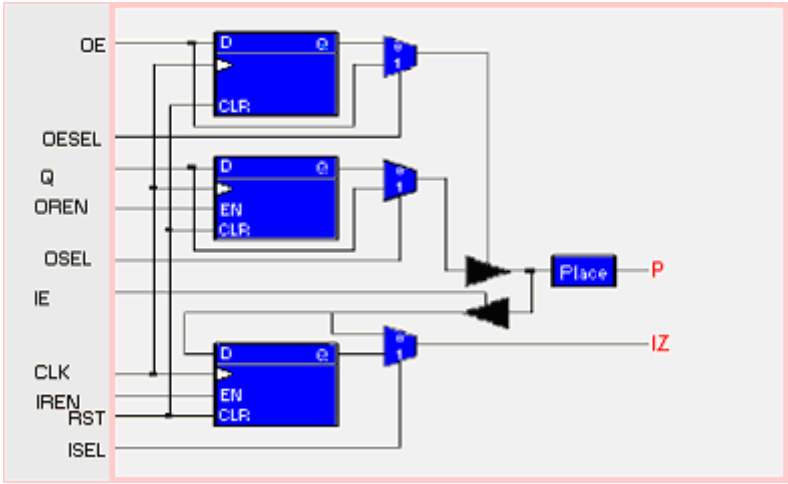
Port Name	Net Connected	Description
P[3]	GND	Pull-up programmable drive strength.
P[2]	VCC	Pull-up programmable drive strength.
SLEW[1]	VCC	2-bit slew rate control connecting directly to SLEW [1:0] of the GPIO driver.

I/O Configuration Settings

Parameter	Value	Description
VCCIO	3.3	VCC I/O voltage.
I/O Standard	CMOS	I/O bank standard.
Slew Rate	WOW	Slew rate control setting.
Drive Strength P	4	Pull-up programmable drive strength.
Pull-Down	False	Pull-down enabled.
Pull-Up	False	Pull-up enabled.

© 2008 QuickLogic Corporation

Graphic View



Clock Pin

Clock Pin Name	IO53
Cell Type	CLOCK
Clock Type	GLOBAL CLOCK

Net

Net Name	N_7
Fan Out	2
Driver Pin	IO91
Driver Port	INZ
Gate	I8

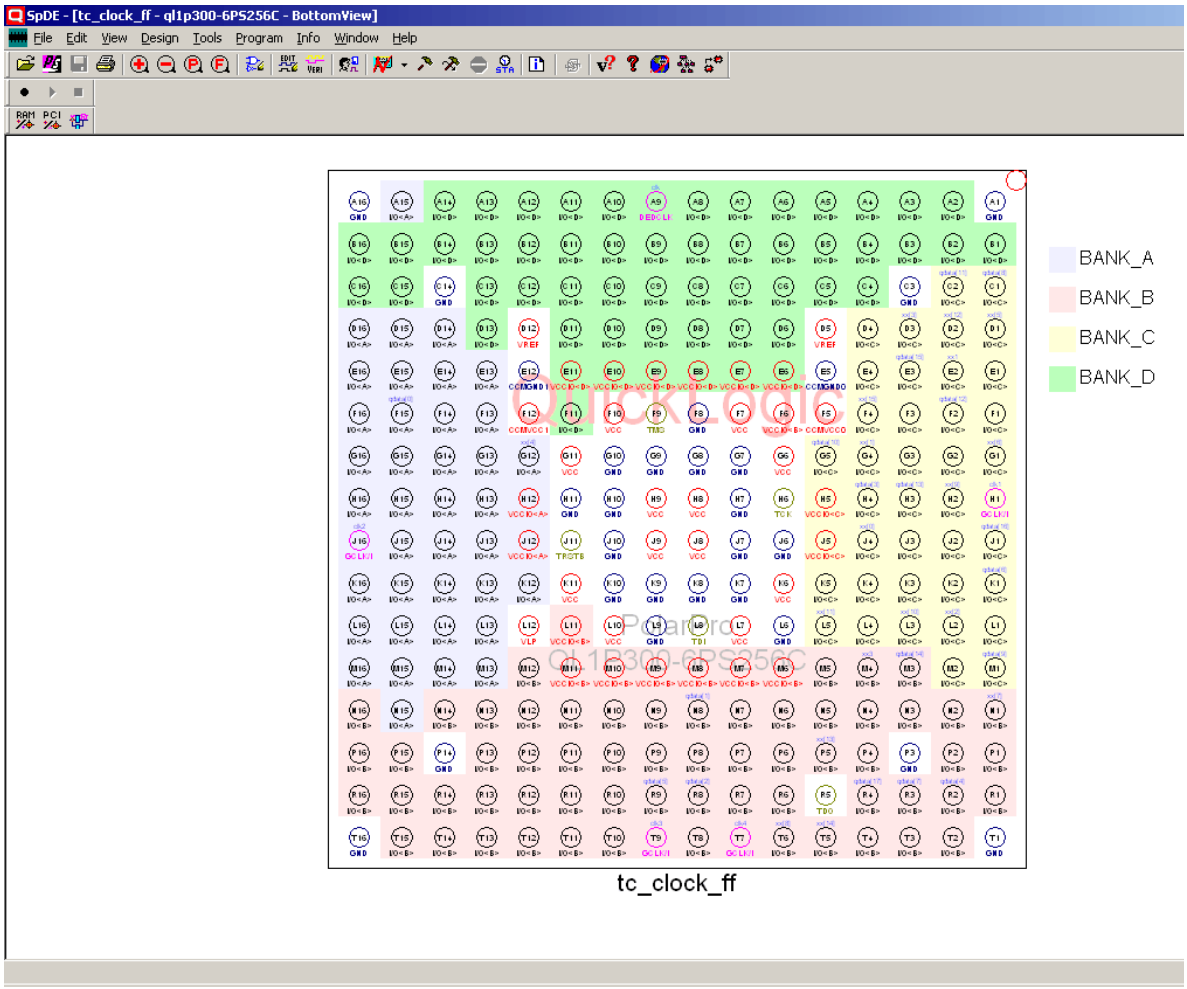
Gate

Gate Name	I2.I1
Gate Type	GPIO
Cell Name	IO132
Fixed	NO
UserFixed	NO

© 2008 QuickLogic Corporation

25.2.3 Package Viewer

This view can be obtained through the menu option **View>Package View**. It displays the selected package and the bonding of pins of package with the I/O pads of chip. Pins of the same I/O bank are displayed in the same color. The color index shows the colors that are assigned to each bank.



25.2.4 Viewer Features

Physical and package view can be displayed in different sizes. The user can zoom in and out using menu options in the **View** menu.
For all options available in sizing the view, refer to the **View** menu help. The complete design can also be viewed in a single window.

While doing fix placement, the user can drag and drop a cell onto the viewer at specific location. This is the convenient way to do fix placement, as the user does not have to type the location in the Fix Placement editor. The physical viewer can be used to fix-place all types of cells, while package view can be used for I/O Pin Placement only.
For more details about fix placement, refer to the Constraint Manager help .

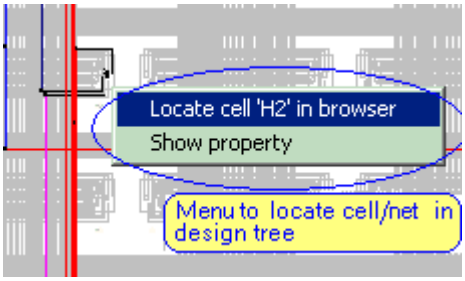
When the splitter option in the physical viewer is used, a two-pane dialog box view is available. To split the viewer, move the mouse pointer to extreme right side of the viewer dialog box. The mouse pointer changes from arrow to splitter. Click, hold and drag the pointer to the desired position. The physical viewer should appear in two different windows, each showing the same view.

Tool tips are displayed in the viewer when the mouse pointer is placed over a resource. The tool tip provides a brief information about a resource.

Various print options are available to print the design as seen in viewer. The user can print the complete view or a selection of a view in different scales. The user can also see the print preview.
For more information, refer the help for Print, Print Preview and Print Selection menu options of the **File** menu.

25.2.4.1 Locating a Cell/Net in the Design Browser

Right-clicking on a cell/net in the physical viewer displays a menu with an option to locate the cell/net in the design tree if it exists. There is also an option to display the properties of the selected cell/net in the property dialog.



25.3 Design Verifier

When a QDIF, EDIF, Verilog, or VHDL file is first open or imported (**File>Import**) into SpDE, the Design Verifier looks for potential design problems. If any design problems are found, they are reported in the transcript window as Verifier Notes, Warnings and Errors found in the design. **Appendix D, "Error Messages" on page 529** provides additional information about these messages.

25.3.1 Types of Errors in the Design Verifier

Notes: Provides potentially important information about the design, such as unused gates that are being removed. Notes are displayed in the *Verifier Notes* tab of the Transcript window.

Warnings: Provides potentially important information about the design, such as high fanout. Warnings are displayed in the *Errors and Warnings* tab of the Transcript window in blue color.

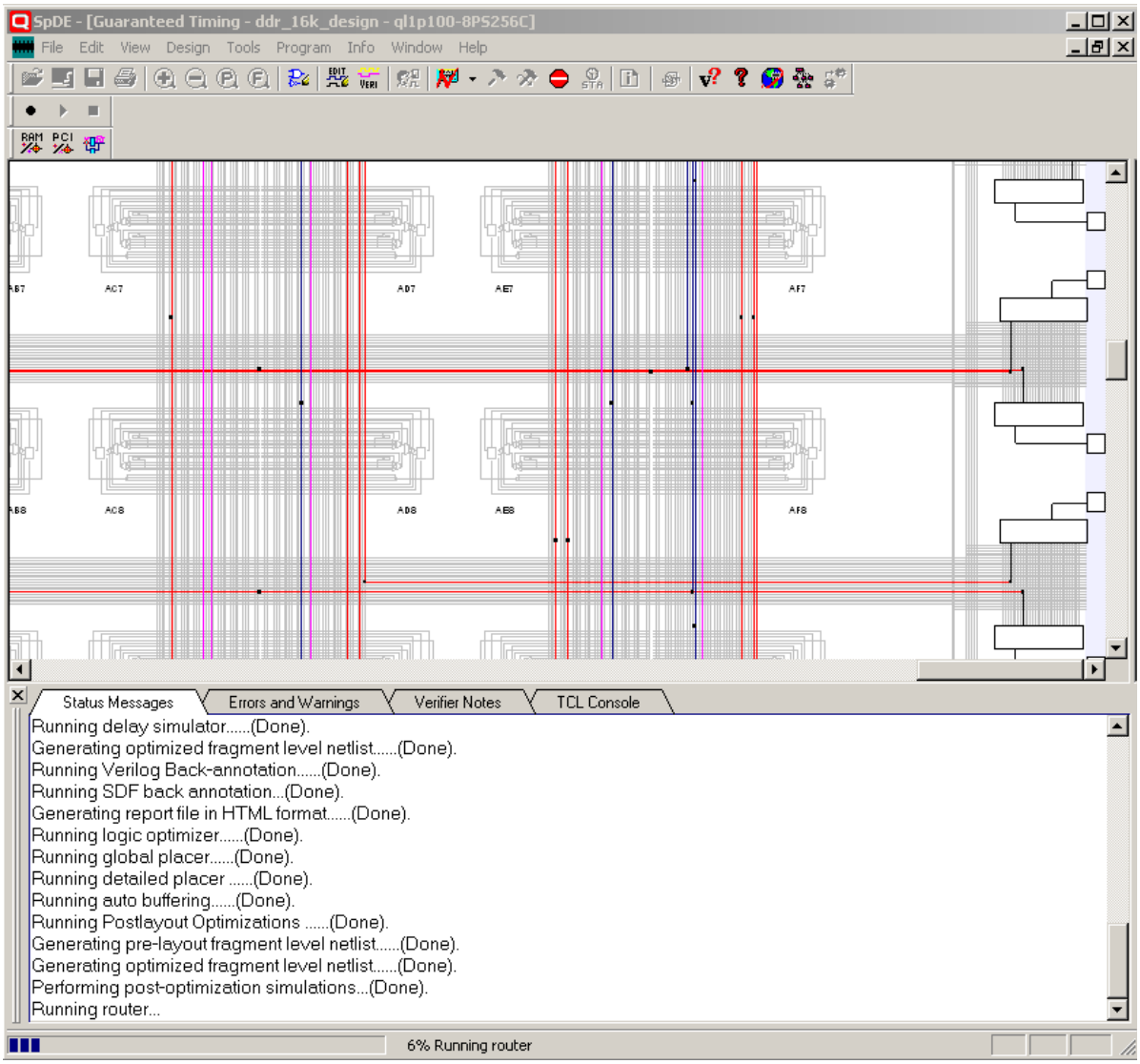
Errors: Provides information about design problems that will prevent a part from being programmed, such as unused inputs (but tools still run). Errors are displayed in the *Errors and Warnings* tab of the Transcript window in red color.

© 2008 QuickLogic Corporation

Fatal Errors: Provides information about extreme errors that prevent even the tools from being run on the design, such as the use of too many of a limited resource or illegally connecting gates. Fatal errors are displayed in the *Errors and Warnings* tab of the Transcript window in red color.

25.4 Transcript Window

SpDE opens the transcript window to display the messages of loading a design, running tools and various other activities in SpDE.



25.4.1 Docking Modes of Transcript Window

Docking windows have two display modes: floating or docked. In floating mode, a window has a thin title bar and can appear anywhere on the screen. A floating window is always on top of all other windows.

In docked mode, a window is fixed to a dock along any of the four borders of the main SpDE window.

To show or hide a transcript window:

1. Toggle between show and hide by selecting **View>Log Window** or press **F2**.
2. To hide the window, click the **Hide** button in the corner of the window. This button may be at the upper-left or upper-right corner, depending on the position of the window.

To change a docked window to a floating window:

- Double-click on the gripper of the transcript window.

To dock a floating window:

- Double-click the window title bar to return the window to its previous docked location, or point to the title bar and drag the window to a different dock area.

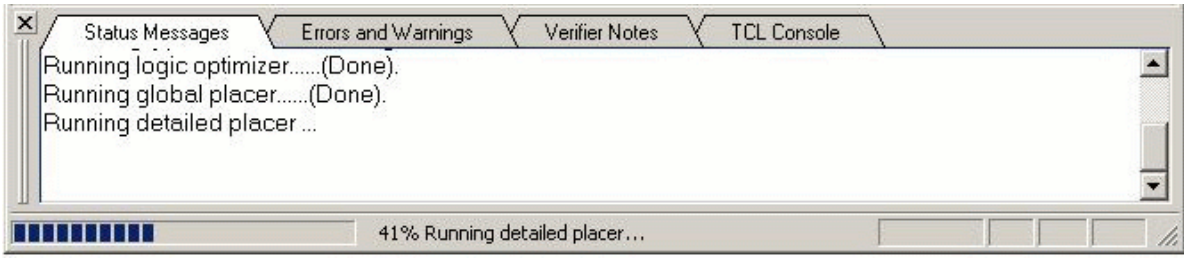
NOTE: The window positions for docking windows are not associated with the current design; they remain the same no matter what design is open.

25.4.2 Transcript Window Tabs

The Transcript window has three tabs that are described in the following subsections. When there is a new message, the tab title blinks for few seconds to indicate to the designer that there is a new message.

25.4.2.1 Status Messages

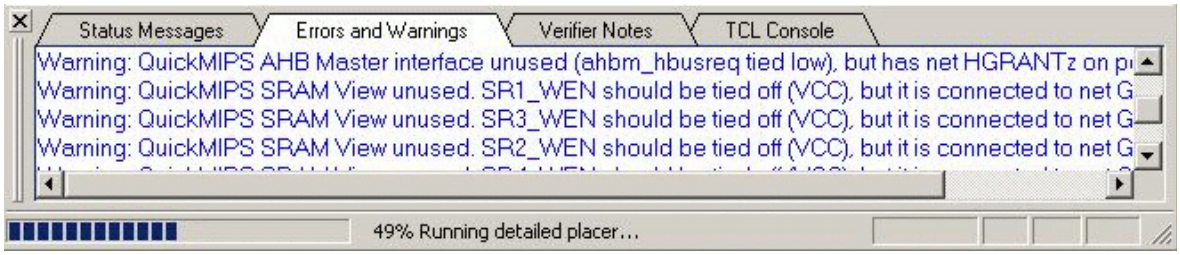
Messages in this tab are displayed for status of loading, run tools, and other designer interactions such as generate report, power calculator, etc.



© 2008 QuickLogic Corporation

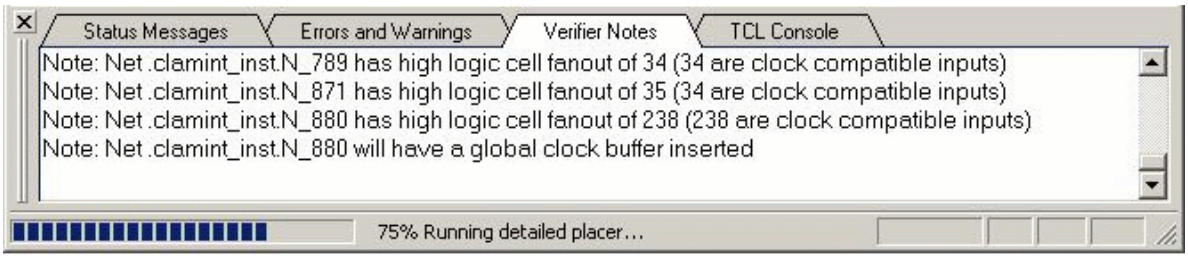
25.4.2.2 Errors and Warnings

Errors and warnings are displayed in this tab. The errors are displayed in red color and the warnings in dark blue and light blue colors. Dark blue indicates high severity warnings and light blue indicates low severity warnings. Errors and warnings include errors and warnings of the verifier.


Status Messages
Errors and Warnings
Verifier Notes
TCL Console
Warning: QuickMIPS AHB Master interface unused (ahbm_hbusreq tied low), but has net HGRANTz on p
Warning: QuickMIPS SRAM View unused. SR1_WEN should be tied off (VCC), but it is connected to net G
Warning: QuickMIPS SRAM View unused. SR3_WEN should be tied off (VCC), but it is connected to net G
Warning: QuickMIPS SRAM View unused. SR2_WEN should be tied off (VCC), but it is connected to net G
49% Running detailed placer...

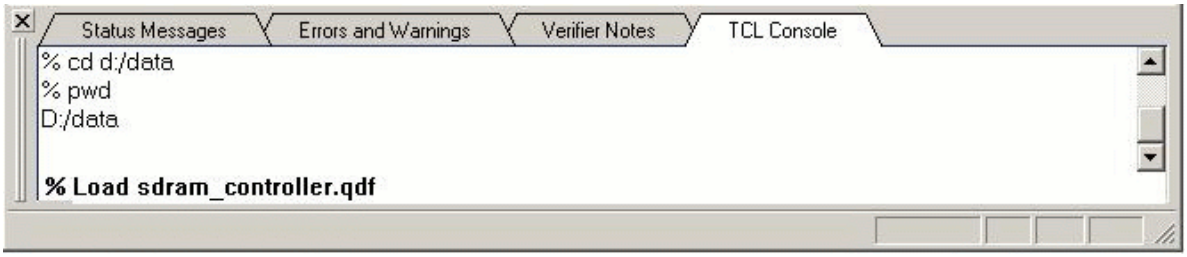
25.4.2.3 Verifier Notes

Verifier messages of the loaded design are displayed in this tab.


Status Messages
Errors and Warnings
Verifier Notes
TCL Console
Note: Net .clamint_inst.N_789 has high logic cell fanout of 34 (34 are clock compatible inputs)
Note: Net .clamint_inst.N_871 has high logic cell fanout of 35 (34 are clock compatible inputs)
Note: Net .clamint_inst.N_880 has high logic cell fanout of 238 (238 are clock compatible inputs)
Note: Net .clamint_inst.N_880 will have a global clock buffer inserted
75% Running detailed placer...

25.4.2.4 TCL Console

TCL commands can be entered at the prompt provided on the TCL Console tab of SpDE's transcript window.



For more information, see also:

- **"TCL Commands for GUI" on page 36**

25.4.3 Progress Bar

The progress bar, which is displayed when loading or run tools is in progress, shows the percentage complete of the progress of the task. Use the **Cancel** button to cancel any task such as loading of the design or run tools in progress.

NOTE: The transcript window is also called *log window*, *output window*. or *message window*.

25.5 File Menu

The File menu contains the commands used for loading, saving, importing, exporting, and other file-related activities. For more information on file types, see the appendix **"File Extensions Reference" on page 389**.

Table 25-1: File Menu Command

File >	Function
Open	Opens a chip file (.CHP), QDIF file (.QDF) or EDIF file (.EDF). Used to open a design into SpDE.
Save	Saves a chip file to the default name (.CHP). Used to save a placed and routed chip to disk.
Save As	Saves a chip file to a specified name. Used when you want to save a chip file to a different name.
Close	Close a chip file (.CHP) or a QDIF file (.QDF). Used to close a design in SpDE.
Save LOF	Saves a chip file to Link Object File (.LOF). In order to Save LOF file, need to first run the sequencer and save the .chp file.
Import Using Precision RTL	Imports: Precision RTL project file (.PSP), Verilog(.V), or VHDL(.VHD) file. Used to load a design into SpDE for the first time. The .PRQ file contains previous settings for Precision RTL. Verilog and VHDL are language based design files.
Print	Prints the current design. Prints the Physical View of the chip on the selected printer to a specified scale.
Print Preview	Previews the printout of the current design.
Print Setup	Sets up the Printer Options. Allows selection of a different default printer or other printer options.
Print Selection	Prints the selected portion of design. The selected portion is shown in Print Preview; from there it can be printed.
Exit	Exits SpDE.

© 2008 QuickLogic Corporation

25.6 View Menu

The View Menu contains commands for manipulating the physical view of the logic cells and routing of a device.

Table 25-2: View Menu Commands

View>	Function
Zoom In	Increases magnification of the Physical View.
Zoom Out	Decreases magnification of the Physical View.
Pan	Pans around the Physical View.
Center	Centers the Physical View.
Full Fit	Shows the entire chip within the SpDE dialog box.
Normal Fit	Zooms to a normal level of magnification.
Lock View	Locks the Physical View to full fit.
Refresh View	Redraws the current chip display. This is useful if the display becomes muddled for some reason.
Go To...	Jumps to a specified Gate, Logic cell, Net or Pad in the Physical View.
Highlight Net	Enters Highlight Net mode. See “Design Constraints and Analysis” on page 287. for more details.
Show Clock Network Usage	Opens the Clock Network Usage tool which highlights all of the nets used on the clock network in the viewer for a given clock input net.
Log Window	Transcript window used to show various messages (errors, warnings, verifier).
Verifier Messages	Displays the verifier warnings and errors in the Transcript Window. See the Design Verifier section for more details.
Package View	Opens a package view of the current design, showing pin assignments.
Design Browser	Displays the Design Browser if it is not already displayed.
Preferences	Allows changes to various options for SpDE. See the descriptions that follow.

The **View>Preferences** command opens the Preferences dialog box to configure the SpDE display and file generation options.

© 2008 QuickLogic Corporation

25.6.1 Texting Tab

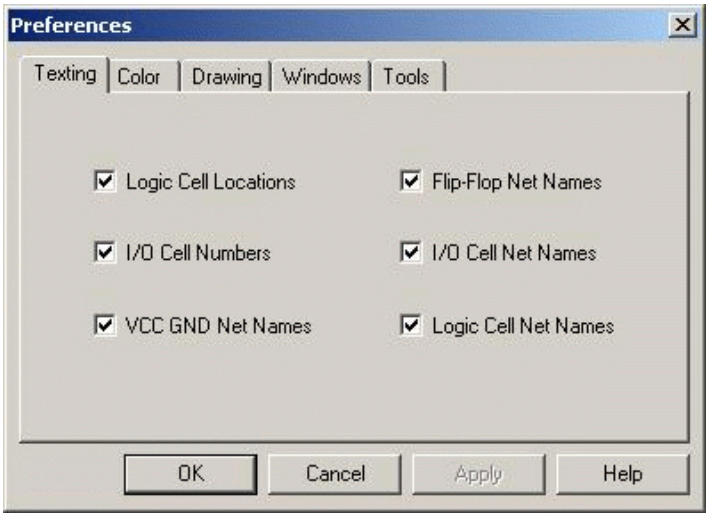


Table 25-3 describes the Texting tab options.

Table 25-3: Texting Options

Texting Option	Function
Logic Cell Locations	Turns on/off display of logic cell locations.
I/O Cell Numbers	Turns on/off display of pin numbers.
VCC GND Net Names	Turns on/off display of VCC and GND net names.
Flip-Flop Net Names	Turns on/off display of flip-flop output net names.
I/O Cell Net Names	Turns on/off display of I/O cell net names.
Logic Cell Net Names	Turns on/off display of all net names.

© 2008 QuickLogic Corporation

25.6.2 Color Tab

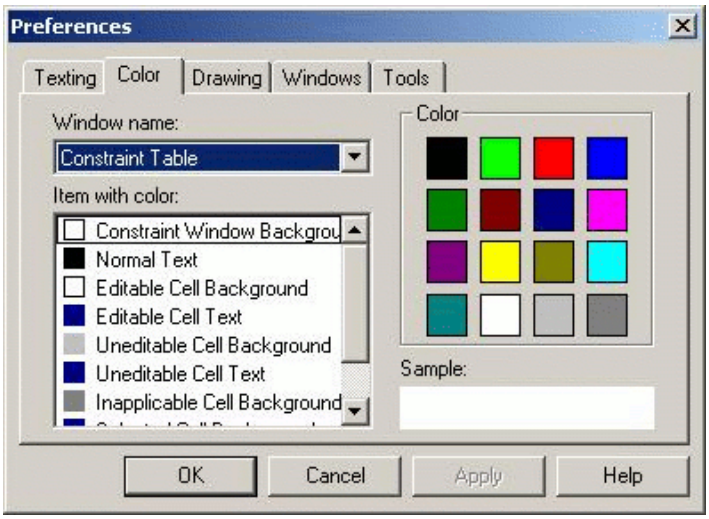


Table 25-4 describes the Color tab options.

Table 25-4: Color Options

Color Option	Function
Constraint Table	Edits colors used in the Constraint Editor.
Internal cells	Edits colors used in the Physical Viewer.
Package Window	Edits colors used in the Package Viewer.
Static Timing Analyzer Graph	Edits colors used in the Static Timing Analyzer graphs.
Static Timing Analyzer Table	Edits colors used in the Static Timing Analyzer results table.

© 2008 QuickLogic Corporation

25.6.3 Drawing Tab

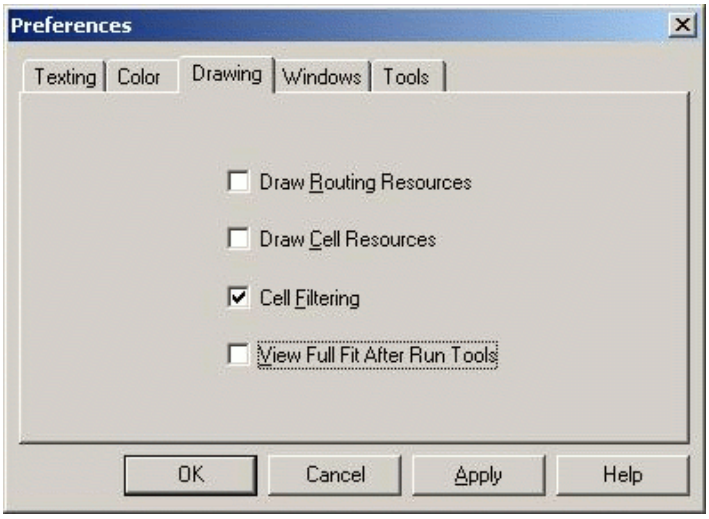


Table 25-5 describes the Drawing tab options.

Table 25-5: Drawing Options

Drawing Option	Function
Draw Routing Resources	Turns on/off display of unused routing.
Draw Cell Resources	Turns on/off display of unused logic.
Cell Filtering	Turns on/off display of unused portion of logic cell.
View Full Fit After Run Tools	Turns on/off a full fit of the physical view after running tools.

© 2008 QuickLogic Corporation

25.6.4 Windows Tab

Table 25-6 describes the Windows tab options.

Table 25-6: Windows Options

Windows Option	Function
Show Standard Toolbar	Turns on/off the standard toolbar
Show Recording Toolbar	Turns on/off the recording toolbar
Show Wizards Toolbar	Turns on/off the wizards toolbar
Show QW Design Flow at Startup	Turns on/off QW design flow at startup
Show Logo	Turns on/off the QuickLogic logo display

© 2008 QuickLogic Corporation

25.6.5 Tools Tab

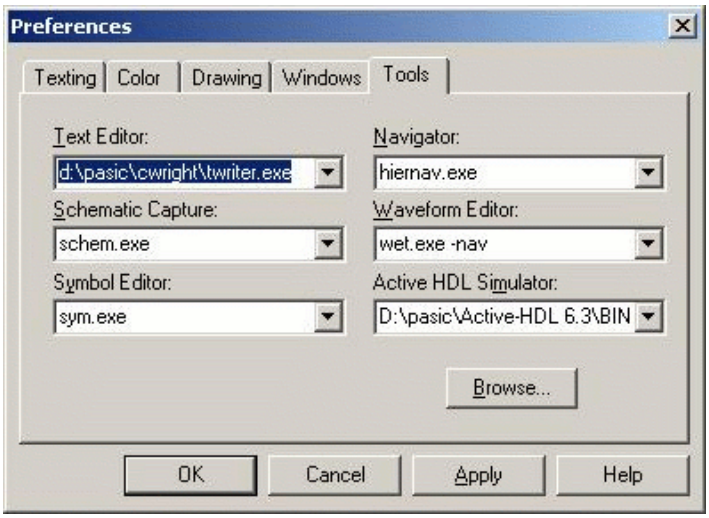


Table 25-7 describes the Tools tab fields.

Table 25-7: Tools Options

Tools Option	Function
Text Editor	Changes the default text editor.
Schematic Capture	Changes the executable for Schematic Capture.
Symbol Editor	Changes the executable for Symbol Editor.
Navigator	Changes the executable for Navigator.
Waveform Editor	Changes the executable for Waveform Editor.
Simulator	Changes the executable for Simulator.

© 2008 QuickLogic Corporation

25.7 Design Menu

The Design menu gives you access to the design and Simulation tools contained within the QuickWorks QuickTools package. For a description of the Design menu items, refer to **Table 25-8**.

Table 25-8: Design Menu Commands

Design>	Function
Schematic Editor & Navigator	Loads a design (`.tre` or `.sch`) into the Navigator. The default is to look for `.tre` (Navigator Attribute) files. Click NEW in the selection dialog box to open a `.sch` (schematic) file. Selects Hierarchy Navigator where a schematic can be edited and navigated through.
Edit Symbol	Selects an SCS Symbol (`.sym`) to edit.
OrCAD Schematic	Generates an edif file that can be loaded in OrCAD for creating an OLB file.
Generate Package Bitmap	Generates a bitmap file of the package view. This option is available only in package view.
Waveform Editor	Opens the SCS Waveform Editor and the Waveform editor for graphical creation of simulation stimulus files (test fixtures). Used only in QuickWorks.
Aldec HDL Simulator	Launch the Aldec Active HDL Simulator.
Update SCS Schematic	Updates SCS or ECS schematics. Do this to update schematics after changing SpDE versions or after modifying symbols. ECS schematics are upwardly compatible.
Edit SCS ini file	Changes SCS options. Changes options for schematic and symbol capture, and Waveform Editor. Also changes library paths.
Text Editor	Opens Turbo Writer for HDL language entry and for any other text editor specified in **View>Preferences.** Use the **File>Open** command in Turbo Writer to open an existing: QuickBoolean (`.QEQ`), Verilog (`.V`), VHDL (`.VHD`), or other text files.

© 2008 QuickLogic Corporation

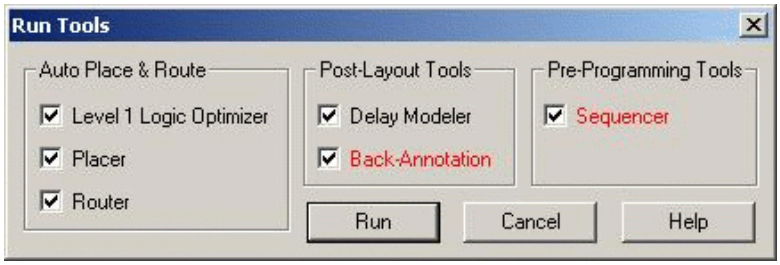
25.8 Tools Menu

The Tools menu is used to set up and run the Logic Optimizer, Placer, Router, Sequencer, Delay Modeler, Back-Annotation, and Path Analyzer tools. In addition, the Tools menu is used to set options for the SpDE tools and the desired simulator. For a description of the Tools menu items, refer to **Table 25-9**.

Table 25-9: Tools Menu Commands

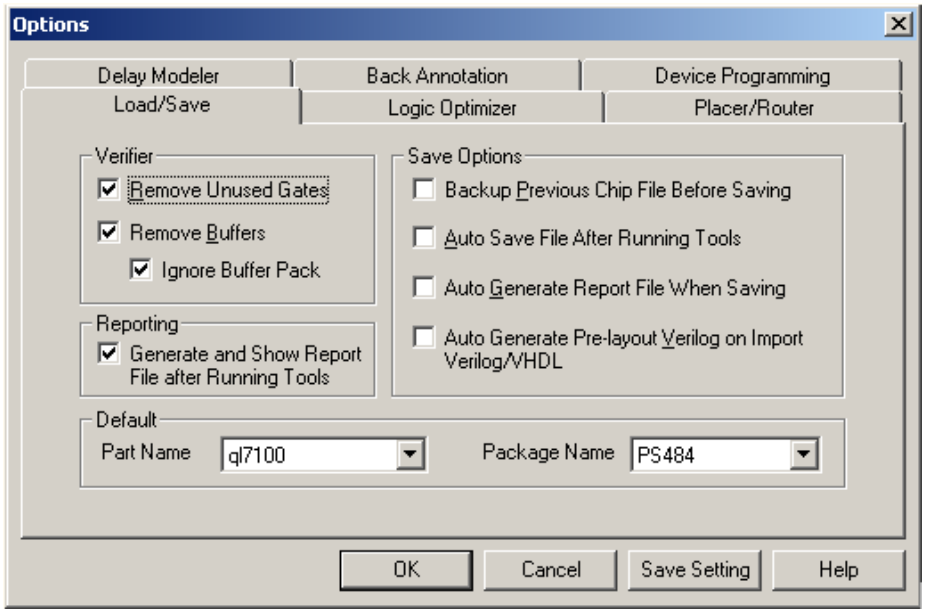
Tools>	Function
Options	Sets up options for SpDE's tools or simulation.
Constraint Manager	Launches Constraint Manager, which is a unified user interface for the design constraint editors.
Run Selected Tools	Selects the tools to run on the design.
Static Timing Analyzer	Opens the Static Timing Analyzer.
Power Calculator	Opens the Power Calculator.
Power Simulator	Launches the Power Simulator tool.
Migrate Part	Launches the Migrate Part tool.
Migrate Part	Launches the Migrate Part tool.
Routing editor	Launches the Manual Routing Editor.
Run TCL Script	Run a TCL Script.
RAM/ROM/FIFO Wizard	Launches the RAM/ROM/FIFO Wizard.
PCI Configuration	Launches the PCI Configuration.
PLL/CCM Wizard	Launches the PLL/CCM Wizard.
Save As Macro	Saving a desing as a macro.

Select **Tools>Run Selected Tools** to open the Run Tools dialog box. This dialog box is used to select which tools are to be run on the design. Disabled tools are grayed-out. The checkboxes for tools that have already been run are cleared. The Sequencer tool is used simply to create data needed to program the device. The Sequencer has no options and is, therefore, not documented. Tools that have not yet been run on the design are in red while tools that have been run are in black.



© 2008 QuickLogic Corporation

Select **Tools>Options** to open the Tools Options dialog box used to configure each of the SpDE tools.



For more information, *see* also:

- **"Using Constraint Manager" on page 268**
- **"Path Analyzer Options" on page 323** - Using Path Analyzer Tool
- **Chapter 18, "Creating RAM, FIFO, and ROM Modules" on page 225** - Using RAM, FIFO, and ROM Modules Wizard
- **"Using Power Calculator" on page 105**
- **Chapter 19, "PCI Configuration for 32-bit Parts" on page 255** - Using 32-bit PCI Configuration Wizard

25.8.1 Load/Save Tab

Table 25-10 lists the options available on the Load/Save tab.

Table 25-10: Load/Save

Load/Save Option	Function
Remove Unused Gates	Turns on/off the removal of unused logic by the Logic Optimizer.
Remove Buffers	Removes buffers from the design being loaded. This is useful when a Synthesis tool inserts buffers non-optimally and the SpDE buffering tool is enabled.
Ignore Buffer Pack	Enabled only when Remove Buffers is selected. If it is selected, the pack attribute on buffers, set by Precision RTL, is ignored while removing them.
Generate and Show Report File after Running Tools	If checked, it automatically generates and shows the report file in the default browser dialog box, after running the tools.

© 2008 QuickLogic Corporation

Table 25-10: Load/Save (Continued)

Load/Save Option	Function
Backup Previous Chip File Before Saving	Creates a backup .chp file of your design before generating a new CHP file.
Auto Save file After Running Tools	Turns on/off automatic save of CHP file after running Tools.
Auto Generate Report File When Saving	Turns on/off automatic generation and saving of the report file when saving a CHP file.
Auto Generate Pre-layout Verilog on Import Verilog/VHDL	Turns on/off automatic generation of Pre-layout Verilog netlist upon import of QDIF.
Part Name	Select the device to place and route the design. The device should be the same as the devices selected for synthesis of design.
Package Name	Select the package of the device to place and route the design. The package of the device should be the same as the package of the device selected for synthesis of design.

NOTE: For detailed descriptions of other Tools Options pages, refer to the following sections:

- **Chapter 12, "Logic Optimizer" on page 137**
- **"Placer Options" on page 147**
- **"Router Tool Options" on page 155**
- **"Delay Modeler Options Tab" on page 172**
- **"Back Annotation Options" on page 174**

25.8.2 Migrate Part Tool

This feature in QuickWorks provides a seamless migration of designs to higher architecture families. For instance, a design in pASIC3 device can be migrated to an Eclipse device, but an Eclipse design cannot be migrated to pASIC3.

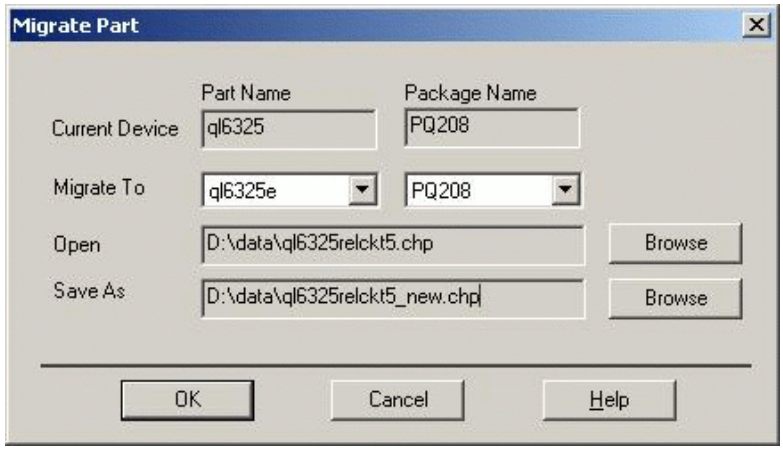
This function is enabled through the **Tools>Migrate Part** menu.

The Migrate Part tool tries to preserve most of the tool run information (like logic optimizer, placer, router, etc.) in the source design. Depending on the resources used in source design, the source design device and the target design device, some or all of the tool run information is discarded, excluding the design netlist. Similarly, depending on the source design package and the target package selected, partial or all pin mapping can be preserved.

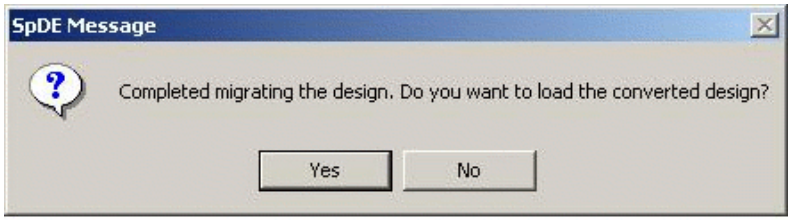
The functionality of this command varies with the device that is read into SpDE. Based on the device that is being migrated, the placement and routing will be preserved appropriately. In some cases, only the placement will be preserved.

© 2008 QuickLogic Corporation

When the Migrate Part is selected, a dialog box similar to the one below, but without any information filled in, opens.



1. To start, type in the source chip file, or click **Browse** to locate the file.
   Part Name and Package Name for the selected `.chp` file are automatically displayed in the Current Device field.
2. Select the target Part Name and Package from the **Migrate To** pull-down menu.
3. Type in the target chip file name or click **Browse** to locate the file.
4. Click **OK**.
   After a successful completion of the operation, the following message appears asking you whether to load the generated chip file.



The output of this command is a new `.chp` file using the new device. When the resulting `.chp` file is loaded into SpDE, the Run Tools is also updated appropriately. For example, in the case when a 7120 design is migrated to an 8325 device, then only the Delay Modeler and Sequencer are displayed as the tools that need to be run. Similarly, in the case of a 7100 design being migrated to an 8250 device, the Router, Delay Modeler and Sequencer should come up as the tools that need to be run.

When the design is migrated across families, e.g., from pASIC3 to Eclipse, in most designs all of the tools need a re-run because the resources available in the new device are different from the source device.

25.9 Program Menu

The Quick*Pro* application is now necessary for programming a part. For information regarding device programming, refer to the *Programmer Kit User's Guide*.

25.10 Info Menu

The Info menu is used to get information about the results of the SpDE tools. A report file that can be generated contains important design information.

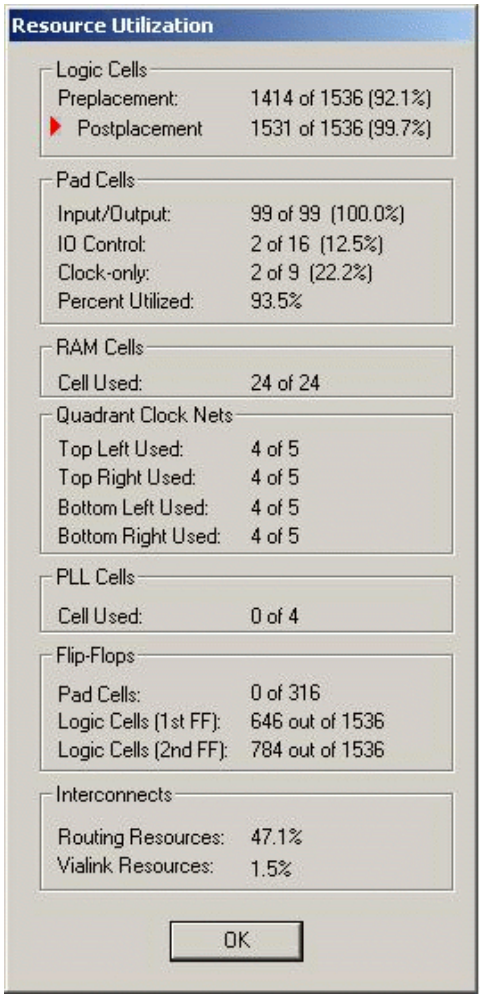

Resource Utilization
Logic Cells
Preplacement: 1414 of 1536 (92.1%)
Postplacement 1531 of 1536 (99.7%)
Pad Cells
Input/Output: 99 of 99 (100.0%)
IO Control: 2 of 16 (12.5%)
Clock-only: 2 of 9 (22.2%)
Percent Utilized: 93.5%
RAM Cells
Cell Used: 24 of 24
Quadrant Clock Nets
Top Left Used: 4 of 5
Top Right Used: 4 of 5
Bottom Left Used: 4 of 5
Bottom Right Used: 4 of 5
PLL Cells
Cell Used: 0 of 4
Flip-Flops
Pad Cells: 0 of 316
Logic Cells (1st FF): 646 out of 1536
Logic Cells (2nd FF): 784 out of 1536
Interconnects
Routing Resources: 47.1%
Vialink Resources: 1.5%
OK

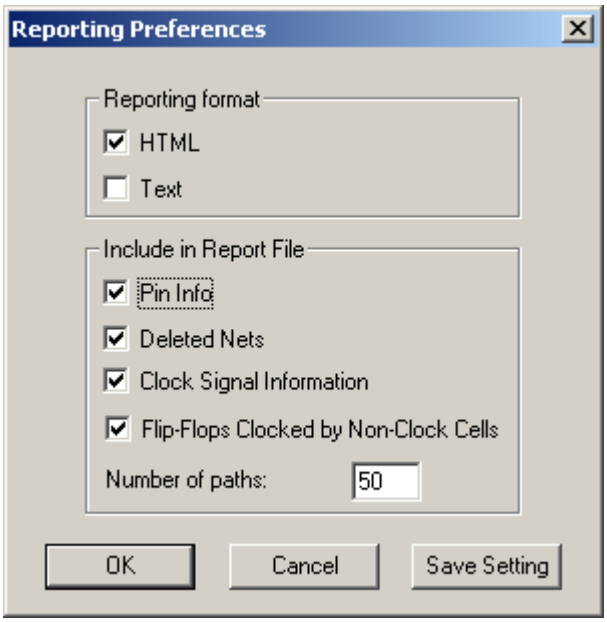
Table 25-11: Info Menu Commands

Info >	Function
Utilization	Displays utilization of logic cells, pad cells, RAM cells, ECU cells, Quadrant clock nets, PLL cells, Flip-flops and Interconnects.
Tool Versions	Lists the tools used to compile the design and versions of tools already used on the design.
Reporting...	Shows Reporting submenu. See **Table 25-12** for details.

© 2008 QuickLogic Corporation

Table 25-12: Reporting Submenu Commands

Reporting >	Function
Report File	Generates a report file in HTML format and displays it in the report view. For details about the Report File, see the *Schematic Design Tutorial* chapter of the *QuickWorks Tutorials User Manual*.
Reporting Preferences	Displays the Reporting Preferences dialog box.



© 2008 QuickLogic Corporation

Chapter 26
Install Devices

This chapter describes the QuickLogic Web Update tool and contains the following sections:

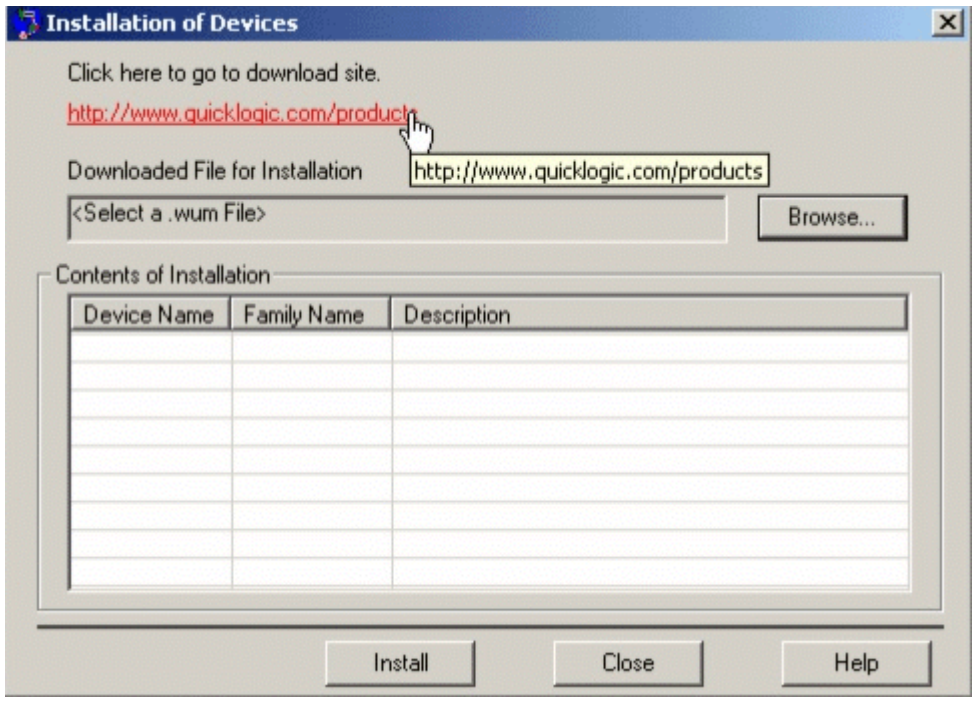
- **"Using Install Devices" on page 385**

26.1 Using Install Devices

This tool provides a web link to download the device updates and new devices. Designers can download the device updates from this web link on their local machines and install them over the local QuickWorks setup.

To begin installation:

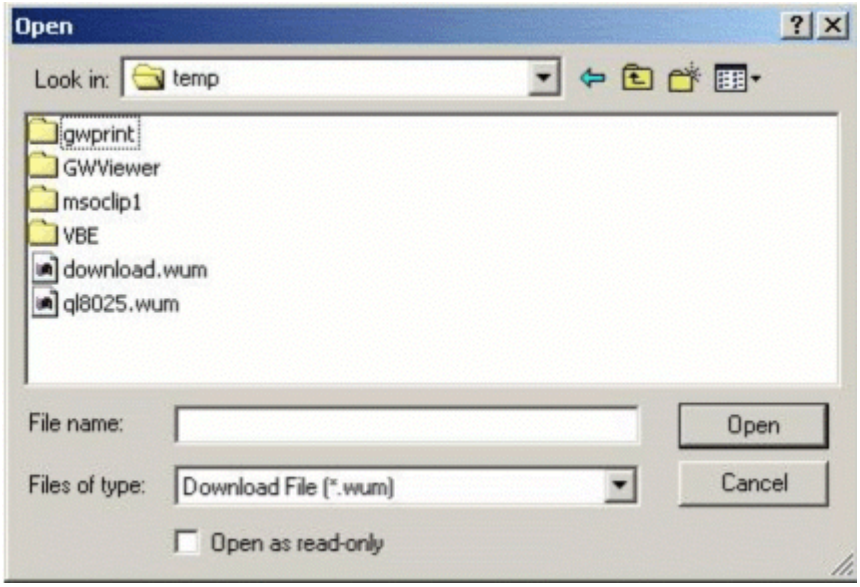
1. Select **Help>Install Devices**.



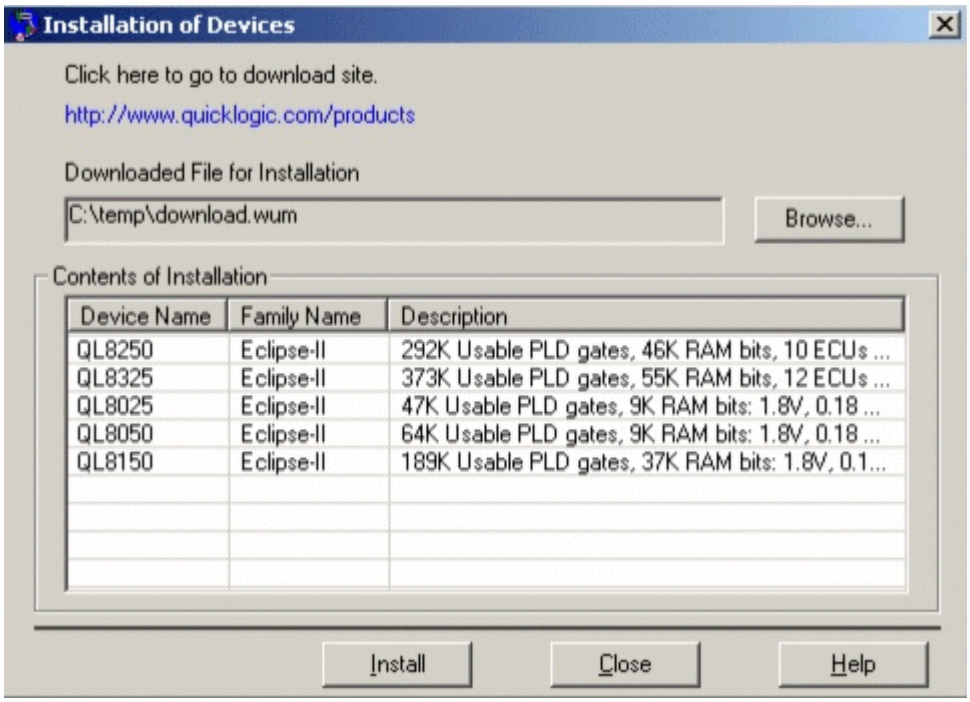
2. Click on the http://www.quicklogic.com link to open the web page for the download.
3. Download the device update files (`*.wum`) from the web to your local machine.

© 2008 QuickLogic Corporation

4. Click **Browse**.
   The Open dialog box is displayed.



5. Select the downloaded file.
6. Click **Open**.
   The contents of the download file are listed.



7. Click **Install** to install the device files.
   If the installation requires overwriting existing files, a list of the changed files are displayed. During the overwriting process the old files are retained by renaming them to *<old name>*_old.*.
8. If the selected file is not correct, a message displays indicating that the file is corruptor that the selected file has an incorrect extension.

© 2008 QuickLogic Corporation

Chapter 27
Web Update

This chapter describes the QuickLogic Web Update tool and contains the following sections:

- **"Web Update at Start Up" on page 387**
- **"Using Web Update" on page 388**

27.1 Web Update at Start Up

When the SpDE application starts, and if there are any updates available for the installed QuickWorks version, a message box appears asking you if you want to download new the new updates.

If **Yes** is clicked, the "Web Update" dialog appears (see **Chapter 27.2** for details).

NOTE: This works only if there is internet connection on your computer.

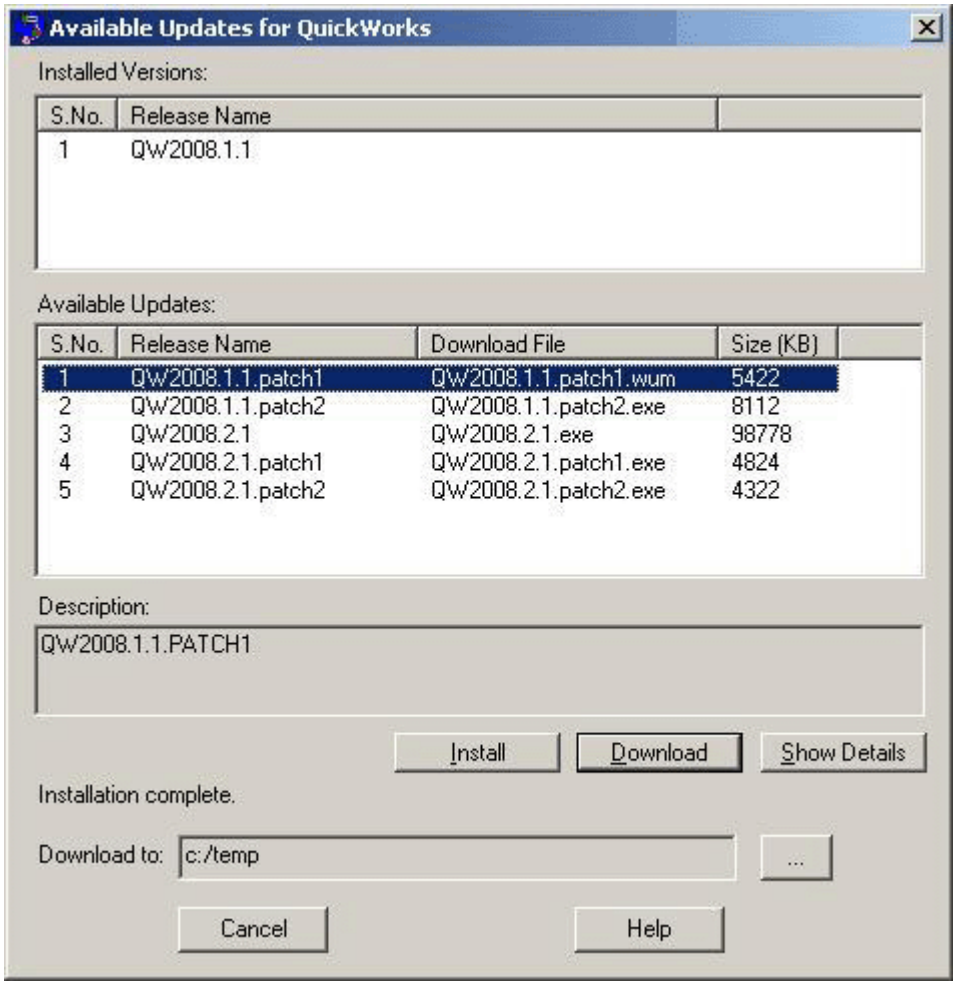
© 2008 QuickLogic Corporation

27.2 Using Web Update

By connecting to the web this tool shows the list of updates available for the installed version of QuickWorks.

To begin update:

1. Select **Help>Web Update** or click on the toolbar.



2. Click **Show Details** to view a "read me" text of the selected release in note pad.
3. Click **Download** to download the selected release in the directory specified in the **Download to:** field, or click **...** (the ellipses button) to browse to the desired download folder.
4. Click **Install** to the download and install the selected wum file. To install non-wum files follow the instructions in the "read me" text of the release.

NOTE: All the QuickWorks applications must be closed to ensure proper installation.

© 2008 QuickLogic Corporation

Appendix A
File Extensions Reference

This appendix provides a complete set of QuickWorks file extensions. It contains the following sections:

- **"File Extensions" on page 389**

A.1 File Extensions

Table A-1: File Extensions

File	Created By	Used By	Description
.ADF	Active HDL	Active HDL	Design description file
.AFC	Active HDL	Active HDL	Drawing file
.ASC	Active HDL	Active HDL	Activ-CAD Test Vectors
.ASF	Active HDL	Active HDL	State diagram source file
.ATR	SpDE Back Annotation	SCS Navigator Back Annotate	A text file describing design attributes.
.AWF	Active HDL	Active HDL	Waveform file
.BAK	Turbo Writer	User	Backup of last edited file.
.BAS	Active HDL	Active HDL	Command Source file
.BV	Back Annotation Tools	Post-Layout Simulation	Back annotated configuration Verilog file.
.CFG	Various	Various	Various configuration files
.CHP	SpDE File-Save	SpDE File-Open	Text file representing placed and routed design.
.CHS	SpDE File-Save	SpDE File-Open	Encrypted file representing placed and routed design.
.CV	Netlister Tools	Pre-Layout Simulation	(1) Configuration Verilog file for schematic designs. (2) For Verilog-only design pci64.cv file to be renamed by user.
.DO	Active HDL	Active HDL	Script file
.DTB	SpDE Back Annotation	Viewlogic VSM Netlister	Delay file for the ViewLogic Netlister.
.EDI	Synopsys, Cadence, ...	SpDE File-Import EDIF	EDIF Netlist File to be imported by SpDE.
.EDIF	Synopsys, Cadence, ...	SpDE File-Import EDIF	EDIF Netlist File to be imported by SpDE on a UNIX platform.
.EDO	SpDE Back Annotation	Various Simulators	EDIF File generated by SpDE to interface with various simulators.

© 2008 QuickLogic Corporation

Table A-1: File Extensions (Continued)

File	Created By	Used By	Description
.EPR	Active HDL	Active HDL	List of source files for compilation
.ERF	Active HDL	Active HDL	compiler errors and messages
.ERR	SCS Design Entry Tools	User	Error file for Consistency Check command in Schematic Editor or Navigator.
.FLW	Active HDL	Active HDL	Flow description file
.KF	SpDE Back Annotation	Intergraph Simulation	QuickLogic delay scaling information.
.LIB	Active HDL	Active HDL	Library index files
.MFG	Active HDL	Active HDL	Library main files
.NEQ	VL2SPDE	SPDE2VL	Text file representing net equivalencies for ViewLogic.
.OID	Active HDL	Active HDL	HDL Wizard auxiliary file
.PL	Active HDL	Active HDL	Perl script file
.PRG	SpDE- Program Part	User	Text file describing programming problems.
.PSP	Precision RTL	Precision RTL	Precision RTL project file.
.QCF	SpDE	SpDE	SpDE constraint file.
.QDF	Various design entry packages	SpDE Import QDIF	QuickLogic's text-based netlist format.
.QSC	Path Analyzer	Path Analyzer	Path Analyzer Script File.
.REP	Precision RTL	User	Precision RTL report file.
.RPT	SpDE Info - Report File	User	Detailed report file based on the design loaded into SpDE.
.SCH	SCS Schematic Editor	Navigator	Binary Schematic File.
.SDF	SpDE Back Annotation	Simulators	Standard Delay Format file for Verilog simulation.
.SDF	Active HDL	Active HDL	Standard delay file
.SPD	SpDE	User	SpDE session log.
.SYM	SCS Symbol Editor, Symbol Creator	Navigator	Binary Symbol file for SCS.
.TB	User, Turbo Writer	VHDL simulator	VHDL test bench.
.TCL	Active HDL	Active HDL	TCL/TK script file
.TF	Waveform Editor, Turbo Writer, User	Verilog Simulator	Verilog Test-Fixture (Input Stimulus).
.TRE	Navigator File-Save	Navigator	Text file to hold attributes assigned in the Navigator.
.V	User, SCS Netlister	Verilog Simulator, Precision RTL	Verilog Design and/or Pre-Layout Simulation File.

© 2008 QuickLogic Corporation

Table A-1: File Extensions (Continued)

File	Created By	Used By	Description
.VH	Precision RTL, SCS Netlister	SpDE Back Annotation tool	Verilog File Header Information (port lists and bus signals).
.VHD	User, SCS Netlister	Precision RTL, VHDL simulator	VHDL pre-layout netlist.
.VHH	SCS Netlister	SpDE - back annotation	VHDL back-annotated header file.
.VHQ	SpDE - back annotation	VHDL simulator	VITAL-VHDL post-layout netlist for use with 3rd party VHDL simulator.
.VL	SpDE Back Annotation	QuickLogic's Viewlogic Interface	Intermediate file for back annotation to the Viewsim Simulator.
.VQ	SpDE-Import VHDL or Back Annotation	Verilog simulator	Structural Verilog Simulation Netlist, Pre-Layout after Import VHDL, Post-Layout after Back Annotation.
.WAV	SCS Netlister Waveform Viewer	Waveform Viewer	Text file listing waveforms to display in the Waveform Viewer for simulation.
.WDL	SCS Netlister Waveform Editor	Waveform Editor	Text file describing input stimulus.
.WET	SCS Netlister Waveform Editor	Waveform Editor	Text file describing the Waveforms on the screen in the Waveform Editor.
.WSP	Active HDL	Active HDL	Environment settings file for a design and workspace.

© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation

Chapter B
Macro Library

This chapter provides an overview of the QuickWorks Macro Library and contains the following sections:

- **“Macro Library Overview” on page 394**
- **“Hard Macros” on page 395**
- **“Soft Macros” on page 395**
- **“I/O Pads” on page 396**
- **“Gates” on page 404**
- **“Latches” on page 416**
- **“Multiplexers” on page 421**
- **“Registers” on page 424**
- **“0.25 um Device Registers” on page 427**
- **“Shift Registers” on page 432**
- **“Up Counters” on page 435**
- **“Down Counters” on page 441**
- **“Up/Down Counters” on page 444**
- **“pASIC Arithmetic” on page 447**
- **“Combinatorial Macros” on page 455**
- **“RAM Blocks” on page 461**
- **“RAM Blocks for 0.25 um Devices” on page 465**
- **“RAM Blocks for PolarPro Devices” on page 468**
- **“RAM Blocks for PolarPro II Devices” on page 469**
- **“PCI Interface Cores Macros” on page 470**
- **“Embedded Computational Units (ECUs)” on page 474**
- **“Low Voltage Differential Signal (LVDS)” on page 482**
- **“Low Voltage Positive Emitter Coupled Logic (LVPECL)” on page 485**
- **“7400-Series TTL” on page 489**
- **“Master Cells” on page 494**
- **“pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro and PolarPro II-Specific Macros” on page 495**
- **“pASIC3, QuickRAM, and QuickPCI-Specific Macros” on page 497**
- **“QuickPCI-Specific Macros” on page 504**
- **“Eclipse-Specific Macros” on page 505**
- **“PolarPro-Specific Macros” on page 512**
- **“PolarPro II-Specific Macros” on page 517**

© 2008 QuickLogic Corporation

B.1 Macro Library Overview

The QuickLogic Macro Library contains over 500 macros and macro building blocks. While these macros offer a wide range of functions and flexibility, they fall into familiar functional groups. The naming conventions employed in the library are easy to learn and remember.

The QuickLogic Macro Library is organized in two main groups: hard macros and soft macros. Hard macros provide simple functionality on a scale of two to twenty gates in complexity, and are no larger than one logic cell in size. Soft macros provide more complex functionality for registers, adders, counters, shifters, etc.

The remarkable flexibility of the pASIC logic cell allows literally millions of different hard macros to be created; not all of these are included in the macro library. Therefore, custom hard macros can be created easily using the pASIC master cell, presented at the end of this chapter.

In general, all the generic symbols in the macro library support pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro, and PolarPro II devices. However, there is a set of macros which allow designers to explicitly utilize pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro, and PolarPro II-specific features such as the Input/Feedback registers and global clock networks (see **“pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro and PolarPro II-Specific Macros” on page 495** for more information).

There is a set of PCI cores macros targeted for QuickPCI devices (see **“QuickPCI-Specific Macros” on page 504** for more information).

Also available is a set of Embedded Computational Units (ECUs) macros targeted for Eclipse devices (see **“Embedded Computational Units (ECUs)” on page 474** for more information).

© 2008 QuickLogic Corporation

B.2 Hard Macros

Hard macros are those macros that can be implemented in a single pASIC logic cell or a fraction of a single pASIC logic cell. These macros provide functions as simple as a 2-input AND gate, or as complex as a dual D latch with enable.

The QuickLogic Macro Library includes `AND, NOR, NAND,` and `OR` gates with two to six inputs. At each input count, all numbers of inversion bubbles are available (for example, 3-input gates are available with `0, 1, 2,` and `3` inversion bubbles). The library also features the largest possible `AND, NOR, NAND,` and `OR` gates which can be implemented in a single logic cell `(AND14i7, NOR14i7, NAND13i6, OR13i6)`.

B.3 Soft Macros

The QuickLogic Macro Library includes a wide variety of soft macros, optimized for speed and density. As the soft macros are built from hard macros, they provide good examples of the use of the macros presented earlier.

© 2008 QuickLogic Corporation

B.4 I/O Pads

The QuickLogic Macro Library has a wide selection of I/O pad macros, including high-drive pads, clock pads, input pads, output pads, tri-state pads, bi-directional pads, and double data rate pads in various configurations.

I/O pads are required in all schematic-based designs. Each QuickLogic device has a different number of pins available. Most pins are standard I/O pins, which use the standard INPAD, OUTPAD, TRIPAD, AND BIPAD pad macros for pASIC3, QuickRAM, QuickPCI, PolarPro, and PolarPro II devices. For Eclipse (0.25 µm) devices, the standard I/O pad macros are INPAD_25um, OUTPAD_25um, TRIPAD_25um, and BIPAD_25um. Please see the pad descriptions in this section for more information.

In earlier releases, the bussed INPADFF (for pASIC3, QuickRAM, and QuickPCI devices) was named INPADxFF (and BPADxFF was called BIPADxFF). For compatibility these older macros are located in the OLDMACRO library. For new designs, the bussed INPADFF (for pASIC3, QuickRAM, QuickPCI, PolarPro, and PolarPro II devices) is named IPADxFF and is located in the PAD library. The bussed INPADFF_25um for Eclipse (0.25 µm) devices is named IPADxFF_25um and is located in the PAD library.

The macro library also includes bussed versions of the INPAD, OUTPAD, TRIPAD, and BIPAD macros targeted for pASIC3, QuickRAM, QuickPCI, PolarPro and PolarPro II devices. These macros are named by appending the number of bits to the pad macro name. The bussed macros are named IPAD4, IPAD8, IPAD16, etc. For Eclipse (0.25 µm) devices, the bussed versions of the INPAD_25um, OUTPAD_25um, TRIPAD_25um, and BIPAD_25um macros are named IPAD4_25um, IPAD8_25um, IPAD16_25um, etc.

All macros supported by the PolarPro and PolarPro II device families follow the same naming convention as those supported by pASIC3, QuickRAM and QuickPCI families (e.g., the input pad INPAD is the same in the PolarPro and PolarPro II families and in the pASIC3, QuickRAM and QuickPCI families, whereas the corresponding pad in the Eclipse family is INPAD_25um).

The PolarPro device family has special *double data rate* (DDR) I/O pads to be used by macros such as ddr_4q, ddr_8q, etc.

Table B-1 on page 396 shows pad macro backward compatibility.

Table B-1: Backward Compatibility

Old Bussed Pad Macro	New Bussed Pad Macros for pASIC3, QuickRAM, QuickPCI, PolarPro and PolarPro II Devices	New Bussed Pad Macros for Eclipse and EclipsePlus (0.25 µm) Devices
INPAD4	IPAD4	IPAD4_25um
BIPAD4	BPAD4	BPAD4_25um
OUTPAD4	OPAD4	OPAD4_25um
TRIPAD4	TPAD4	TPAD4_25um

To use a pad with a different number of bits, create an iterated instance or a custom symbol.

NOTE: I/O pads (INPAD, OUTPAD, TRIPAD, BIPAD, INPAD_25um, OUTPAD_25um, TRIPAD_25um, and BIPAD_25um macros) that are not used in a design are automatically tied to ground within the chip.

© 2008 QuickLogic Corporation

NOTE: If you wish to leave a pin un-driven, then place a TRIPAD or TRIPAD_25um (for Eclipse) in your schematic with the output enable forced to ground (GND). Also, you must put a PLACE attribute on the TRIPAD or TRIPAD_25um indicating which pin-number you want to leave floating. Any TRIPAD or TRIPAD_25um with a fixed pin placement like this will not be removed by the SpDE optimizer.

NOTE: Input-only pads and clock pads (HDPAD, CKPAD, HDPAD_25um, and CKPAD_25um) that are not used in a design are left floating.

B.4.1 CKPAD

CKPADS (Clock Pads) must be used on pins labeled I/ CLK, ACLK/I, or GCLK/I on the pin tables shown in the device appendices. For example, there are:

- Two ACLK/I and two GCLK/I pins on pASIC3 (QL3012 and QL3025), and QuickRAM (QL4016 and QL4036) devices
- Two ACLK/I and six GCLK/I pins on pASIC3 (QL3040 and QL3060), and QuickRAM (QL4058 and QL4090) devices

Clock pads can be placed on any pin labeled ACLK/I (pASIC3, QuickRAM, and QuickPCI), or GCLK/I (pASIC3, QuickRAM, and QuickPCI) in the Device Pinout appendices. The basic clock pad, `CKPAD`, provides access to a clock network. Each of these dedicated resources is capable of clocking, setting, or resetting every flip-flop in the pASIC device. Clock pads can be placed on any pin labeled `ACLK/I` (pASIC3 and QuickRAM), or `GCLK/I` (pASIC3 and QuickRAM) in the Device Pinout appendices.

The `CKdPAD` macro provides a macro compatible with the conventional `HDdPAD` macro. If a clock cell is not required for the clocking, setting, or resetting duty described above, `CKdPAD` allows the clock cell to be used for any purpose.

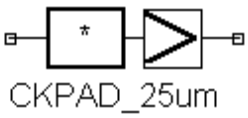
The `CKtPAD` macro provides a union of `CKPAD` and `CKdPAD` functionality. The `CKtPAD` has three output terminals—the center terminal is `CKPAD` equivalent, and therefore can drive only the clock, preset, or clear ports on a flip-flop. The outer pair of terminals are CKdPAD equivalent, and therefore can drive any ports. The `CKtPAD` should be used in designs that require a high fanout common clock, and the clock signal needs to go to logic inputs as well. If a high fanout inverted clock is needed in addition to a high fanout non-inverted clock, the best lowest-skew solution is to invert the clock external to the device, and use the two clock pads, one for the non-inverted clock and one for the inverted clock.

© 2008 QuickLogic Corporation

B.4.2 CKPAD_25um

NOTE: This macro can only be used in Eclipse devices.

Figure B-1: CKPAD_25um

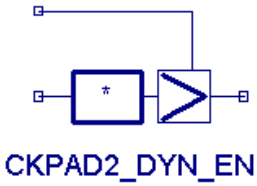


The CKPAD_25um macro is a Eclipse-specific clock pad macro. CKPAD_25um must be used on pins labeled CLK, CLK/ PLLIN, and CLK DEDCLK/ PLLIN on the pin tables shown in the device appendices. There are five CLK, one CLK DEDCLK/ PLLIN and three CLK/ PLLIN pins on Eclipse devices.

B.4.3 CKPAD2_DYN_EN

NOTE: This macro can be used in PolarPro and PolarPro II devices. It is available for all PolarPro devices except QL1P100 and QL1P075.

Figure B-2: CKPAD2_DYN_EN

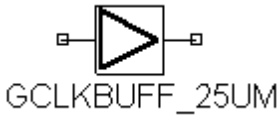


The CKPAD2_DYN_EN macro is a PolarPro and PolarPro II-specific clock pad macro. The enable pin can be used to control the clock usage. CKPAD2_DYN_EN must be used on pins labeled CLK on the pin tables shown in the device appendices. There are five CLK pins on PolarPro and PolarPro II devices.

B.4.4 GCLKBUFF_25um

NOTE: This macro can only be used in Eclipse devices.

Figure B-3: GCLKBUFF_25um



The GCLKBUFF_25um macro is a Eclipse-specific global clock buffer pad macro. GCLKBUFF_25um macro is used to place any internally generated signal on the global clock network. GCLKBUFF_25um must be used on pins labeled CLK on the pin tables shown in the device appendices. There are five CLK pins on Eclipse devices.

© 2008 QuickLogic Corporation

B.4.5 GCLKBUFF

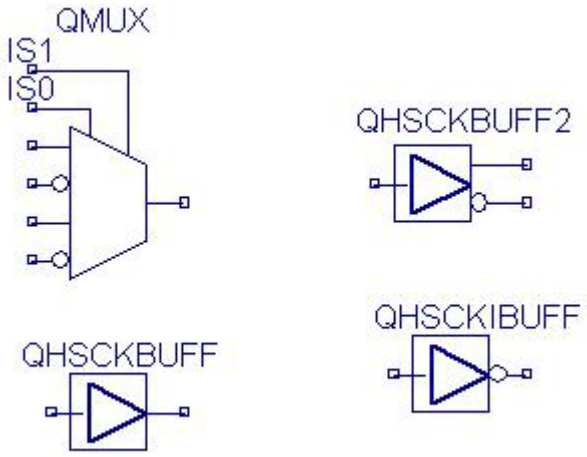
This macro is supported in the PolarPro device family and provides the same functionality as GCLKBUFF_25um.

Figure B-4: GCLKBUFF

B.4.6 QMUX, QHSCKBUFF, QHSCKBUFF2, and QHSCKIBUFF

The QMUX, QHSCKBUFF, QHSCKBUFF2, and QHSCKIBUFF macros are supported in the PolarPro II device family and provide similar functionality as GCLKBUFF for PolarPro. Different macros take inputs from the clock pad, or from the internally generated clock, or from the inverted signals for them.

Figure B-5: QMUX, QHSCKBUFF, QHSCKBUFF2, and QHSCKIBUFF



B.4.7 HDPAD

Figure B-6: HDPAD

High Drive pads (HDPADs) must be used on pins labeled `I` only in the pinout tables shown in the device appendices. There are four of these pads on pASIC3 and QuickRAM devices.

High Drive pads are an important resource in the pASIC architecture, providing twice the drive current of the normal input pad. Furthermore, the High Drive pads may be used to achieve extremely fast paths with minimized skew. High Drive pads are placed on input cells. The Device Pinout appendices show locations where High Drive pads should be used with an `I` (Input-Only).

© 2008 QuickLogic Corporation

The library includes non-inverting and inverting High Drive pads, `HDPAD`, and `HDiPAD` respectively. The latter is employed to invert an active-low reset signal to be compatible with the pASIC flip-flops active-high preset and clear. A dual high-drive pad, `HDdPAD`, provides both non-inverting and inverting outputs.

For high-speed designs refer to the buffering techniques in the *Design Techniques* chapter.

B.4.8 HDPAD_25um

NOTE: This macro can only be used in Eclipse devices.

Figure B-7: HDPAD_25um

The HDPAD_25um macro is a Eclipse-specific high drive pad macro. There are 16 high drive networks in the Eclipse devices. Each bank of I/O has two input-only pins that can be used to drive these high drive networks.

B.4.9 IO_BUFF_25um

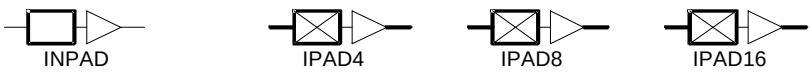
NOTE: This macro can only be used in Eclipse devices

Figure B-8: IO_BUFF_25um

The IO_BUFF_25um is a Eclipse-specific I/O buffer pad macro. IO_BUFF_25um pad macro is used to place any internally generated signal on the high-drive network.

B.4.10 INPAD

Figure B-9: INPADS



`INPADs` should be used on pins labeled `I/O` in the pinout tables shown in the device appendices.

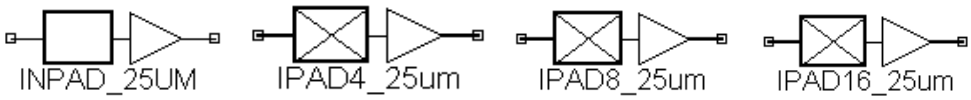
The I/O macro `INPAD` provides a non-inverting input pad with normal drive current.

© 2008 QuickLogic Corporation

B.4.11 INPAD_25um

NOTE: This macro can only be used in Eclipse devices.

Figure B-10: INPAD_25um



The INPAD_25um(s) macros are Eclipse-specific input pads macros. INPAD_25um(s) should be used on pins labeled I/O in the pinout tables shown in the device appendices.

The I/O macro INPAD_25um provides a non-inverting input pad with normal drive current.

B.4.12 OUTPAD

`OUTPADs` should be used on pins labeled `I/O` in the pinout tables shown in the device appendices.

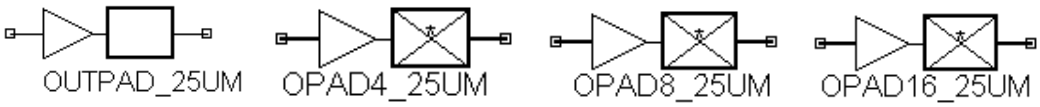
Figure B-11: OUTPADS

The I/O macros `OUTPAD` and `OUTiPAD` provide non-inverting and inverting output pads, respectively. The inverting pad can be used to match off-chip polarity requirements without using an additional inverter macro. The `OUTorPAD` macro adds OR gate functionality to the output pad.

B.4.13 OUTPAD_25um

NOTE: This macro can only be used in Eclipse devices.

Figure B-12: OUTPAD_25um



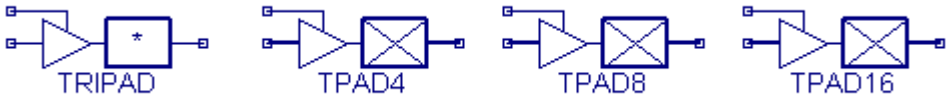
OUTPAD_25um(s) macros are Eclipse-specific output pads macros. OUTPAD_25um(s) should be used on pins labeled I/O in the pinout tables shown in the device appendices.

© 2008 QuickLogic Corporation

B.4.14 TRIPAD

TRIPADs should be used on pins labeled I/O in the pinout tables shown in the device appendices.

Figure B-13: TRIPAD

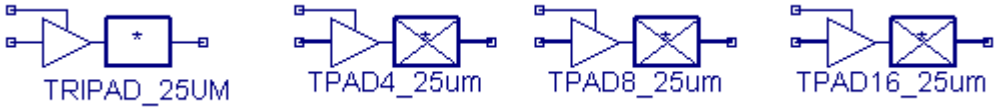


The I/O macros TRIPAD and TRIiPAD provide non-inverting and inverting tri-state pads, respectively. The enable signal on these macros is active-high—when enable is low the pad is in tri-state; when the enable is high the pad is in output-state. The TRIorPAD macro adds OR gate functionality to the tri-state pad.

B.4.15 TRIPAD_25um

NOTE: This macro can only be used in Eclipse-specific devices.

Figure B-14: TRIPAD_25um

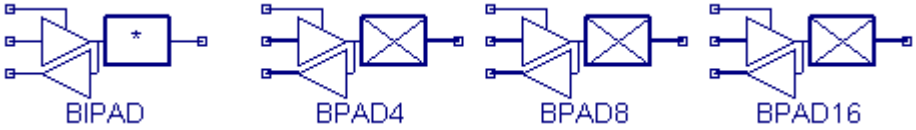


TRIPAD_25um(s) macros are Eclipse-specific tri-state pads macros. TRIPAD_25um(s) should be used on pins labeled I/O in the pinout tables shown in the device appendices.

B.4.16 BIPAD

BIPADs should be used on pins labeled I/O in the pinout tables shown in the device appendices.

Figure B-15: BIPAD and BIiPAD

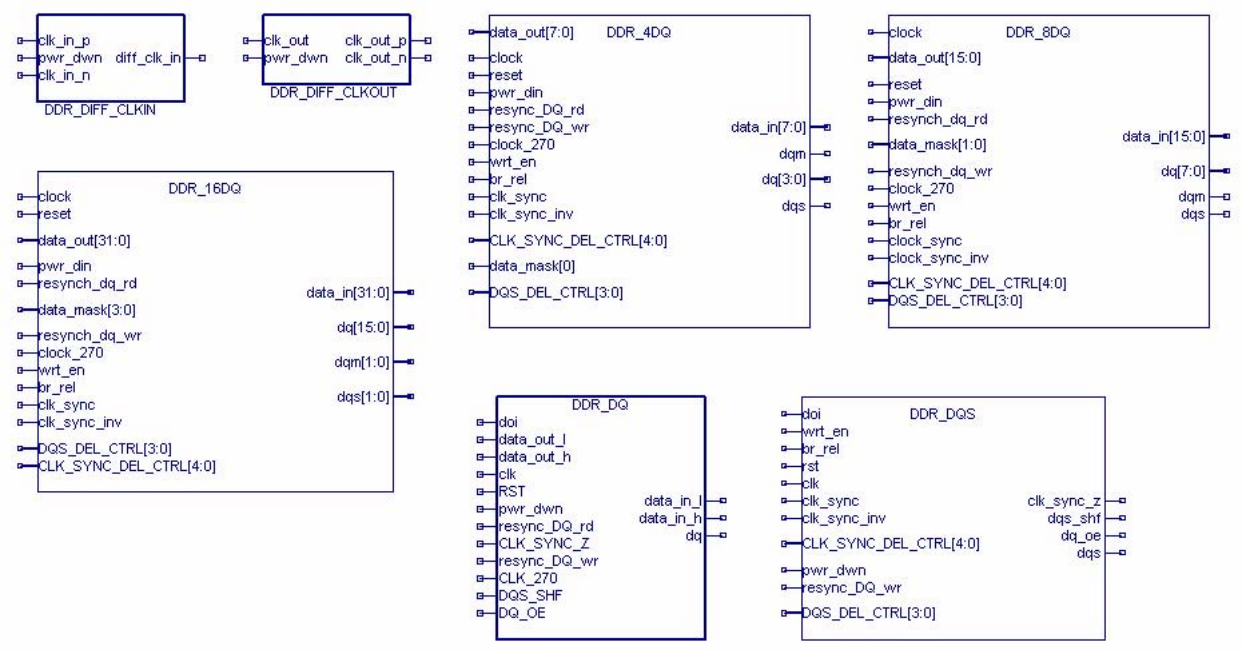


The I/O macros BIPAD and BIiPAD provide non-inverting and inverting bi-directional pads, respectively. The enable signal on these macros is active-high—when enable is low the pad is in input-state, when the enable is high the pad is in output-state. Finally, the BIorPAD macro adds OR gate functionality to the bi-directional pad.

© 2008 QuickLogic Corporation

B.4.17 DDR_DQ

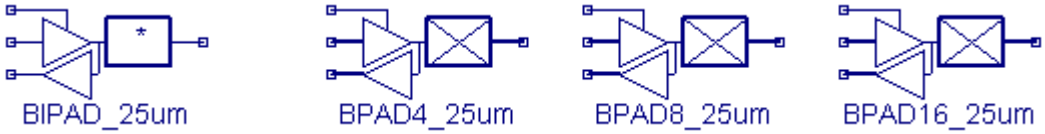
Figure B-16: DDR_DQ Pads



DDR macros are PolarPro-specific pad macros that should be used on pins labeled I/O in the pinout tables shown in the device appendices.

B.4.18 BIPAD_25um

Figure B-17: BIPAD_25um



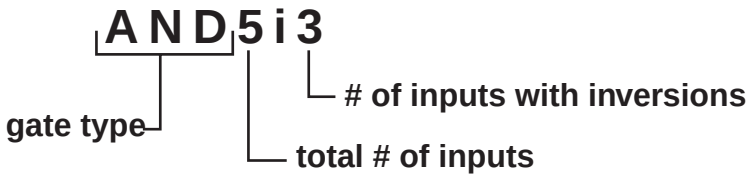
BIPAD_25um(s) macros are Eclipse-specific bi-directional pads macros. BIPAD_25um(s) should be used on pins labeled I/O in the pinout tables shown in the device appendices.

© 2008 QuickLogic Corporation

B.5 Gates

The QuickLogic Macro Library includes `AND, OR, NAND,` and `NOR` gates with two to six inputs with each input invertible. A very wide input gate is available for each gate `(AND14i7, OR13i6, NAND13i6, NOR14i7)`. These special gates are implemented in a single logic cell.

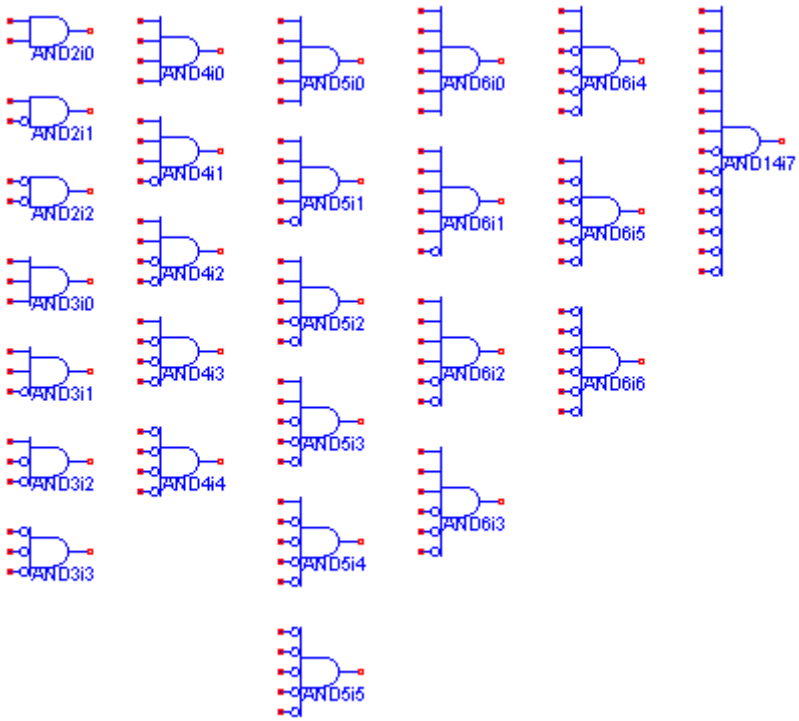
B.5.1 Naming Convention



- The first field identifies the gate type - `AND, OR, NAND, NOR.`
- The second field specifies the total number of inputs.
- The 'i' character separates the fields and indicates inversion.
- The third field specifies the number of inversion bubbles.

B.5.2 AND Gates

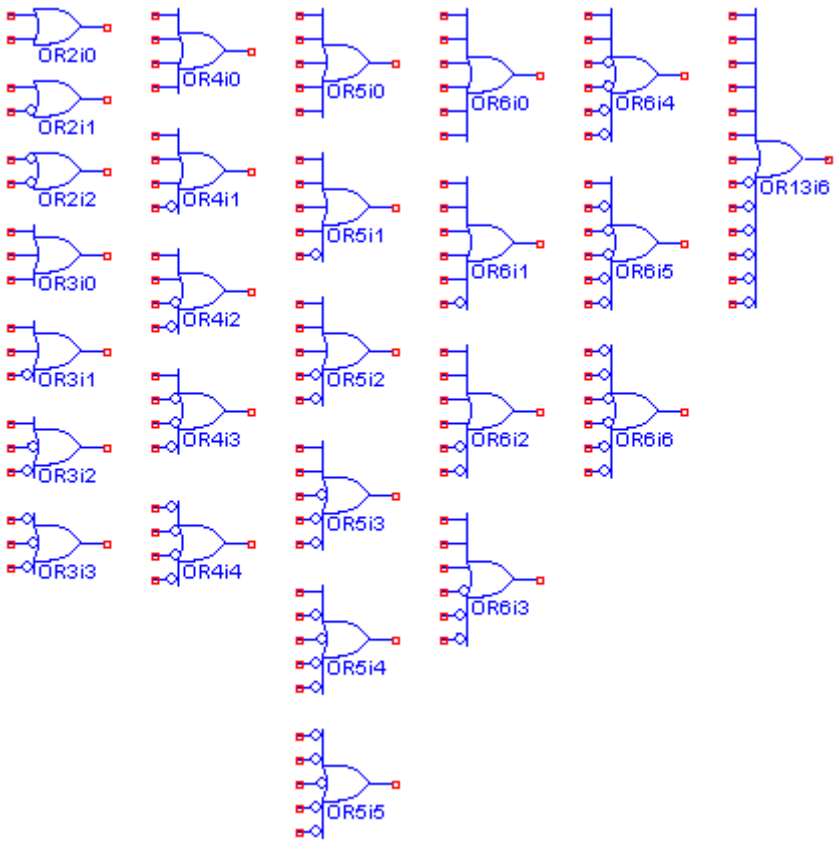
Figure B-18: AND Gates



© 2008 QuickLogic Corporation

B.5.3 OR Gates

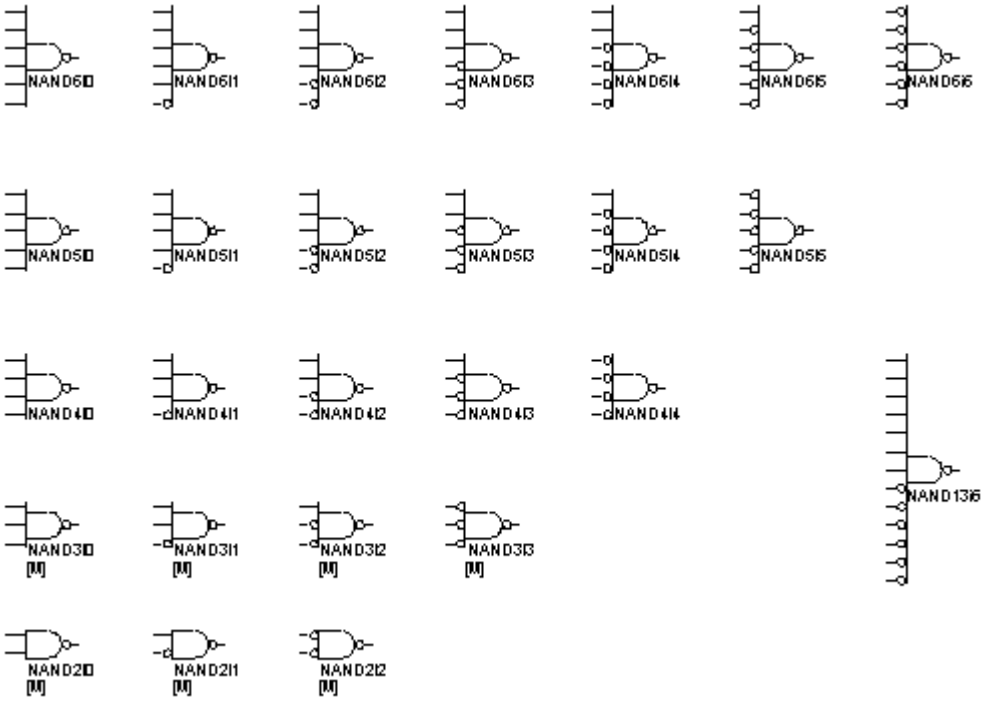
Figure B-19: OR Gates



© 2008 QuickLogic Corporation

B.5.4 NAND Gates

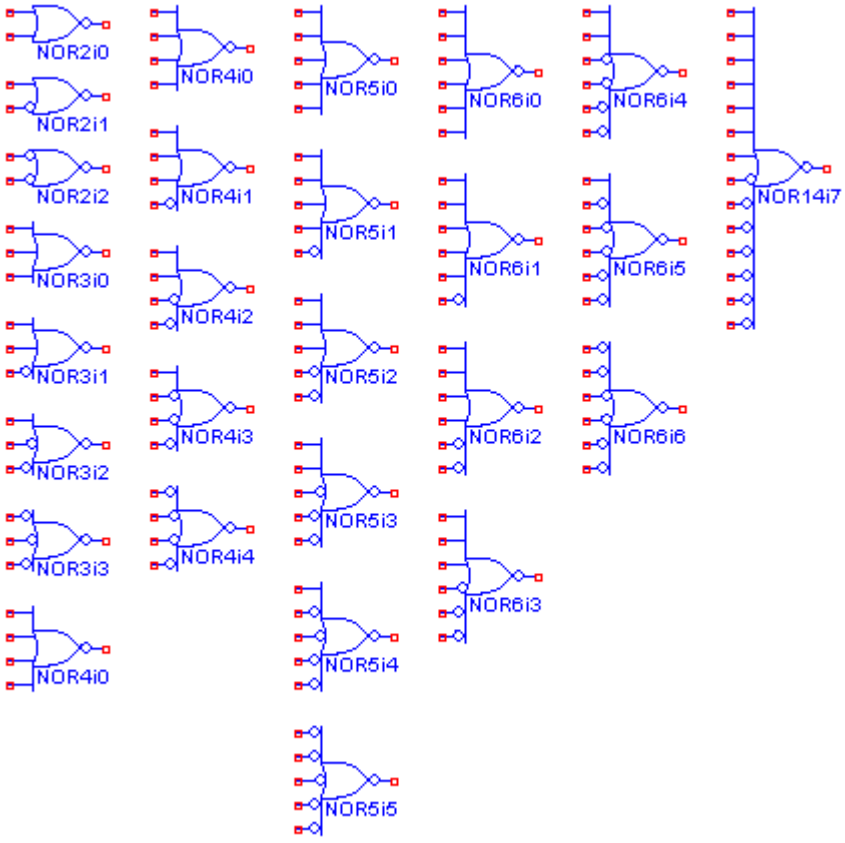
Figure B-20: NAND Gates



© 2008 QuickLogic Corporation

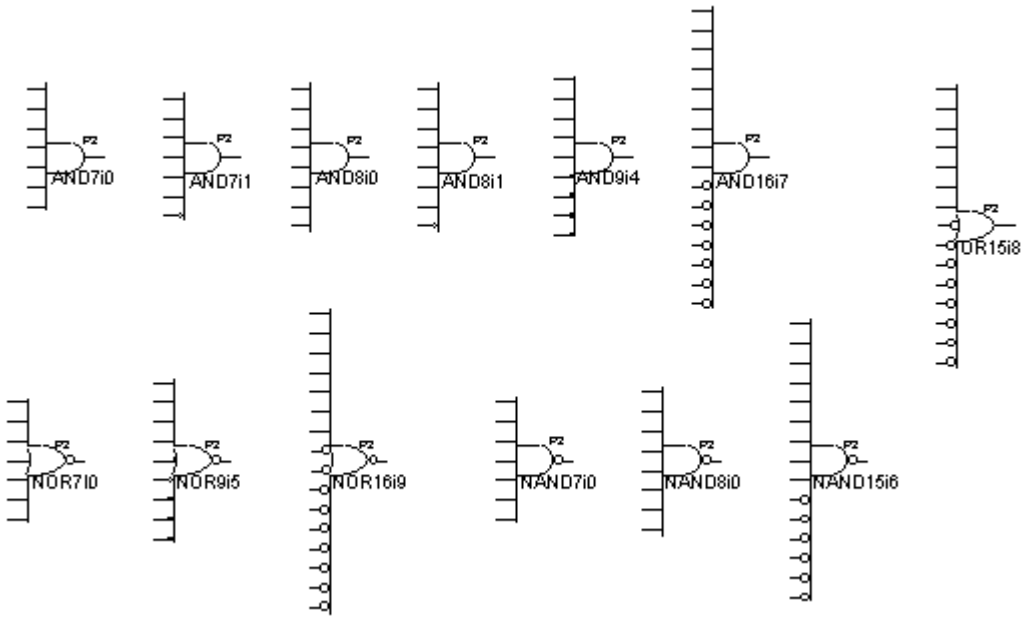
B.5.5 NOR Gates

Figure B-21: NOR Gates



B.5.6 pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro, and PolarPro II-Specific Gates

Figure B-22: pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro and PolarPro II-Specific Gates



The QuickLogic Macro Library has more than a dozen flip-flop macros. DFF is a standard D flip-flop with no frills; TFF is a T flip-flop; and JKFF is a standard JK flip-flop. **Table B-2** lists the features of each flip-flop available in the library. If the *Uses DFF Only* column is marked, then only the dedicated flip-flop in the QuickLogic Logic Cell is used. Otherwise, additional logic resources in the Logic Cell are used.

Table B-2: Flip-Flop Features

Macro	Description	Preset	Clear	Enable	Uses DFF Only
DFF	D flip-flop				x
DFFC	D flip-flop w/ clear		x		x
DFFP	D flip-flop w/ preset	x			x
DFFPC	D flip-flop w/ preset, clear	x	x		x
DFFE	D flip-flop w/ enable			x	
DFFEPC	D flip-flop w/ enable, preset, clear	x	x	x	
DFFsC	D flip-flop with synchronous clear		sync		
TFF	T flip-flop				
TFFPC	T flip-flop w/ preset, clear	x	x		
TFFE	T flip-flop w/ enable			x	
TFFEPC	T flip-flop w/ enable, preset, clear	x	x	x	
JKFF	JK flip-flop				

© 2008 QuickLogic Corporation

Table B-2: Flip-Flop Features (Continued)

Macro	Description	Preset	Clear	Enable	Uses DFF Only
JKFFPC	JK flip-flop w/ preset, clear	x	x		
JKnFFPC	JK flip-flop w/ K inverted, preset, clear	x	x		
JKNN_FF	JK flip-flop w/ K inverted, clear		x		

NOTE: Many flip-flop macros employ the asynchronous preset and clear inherent in each logic cell. Only the DFFsC macro employs a synchronous clear (not to be confused with *asynchronous clear*).

NOTE: The simple D flip-flop macros (DFF, DFFC, DFFP, DFFPC) have preferable density characteristics. These macros are marked in the "Uses DFF Only" column, so they can be implemented without requiring any other extra logic in the pASIC logic cell. These flip-flops will, therefore, result in the densest and fastest circuits.

NOTE: In case of the PolarPro and PolarPro II device families, all DFF macros use only DFF from the logic cell.

B.5.6.1 DFF

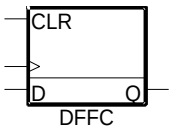
INPUTS: CLK, D

OUTPUTS: Q

Table B-3: DFF

CLK	D	Q
L	X	Q
H	X	Q
^	A	A

B.5.6.2 DFFC



INPUTS: CLK, CLR, D

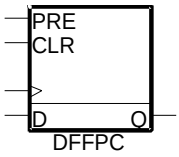
OUTPUTS: Q

Table B-4: DFFC

CLK	CLR	D	Q
X	H	X	L
L	L	X	Q
H	L	X	Q
^	L	A	A

© 2008 QuickLogic Corporation

B.5.6.3 DFFPC



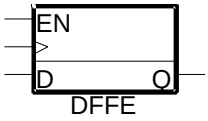
INPUTS: CLK, CLR, PRE, D

OUTPUTS: Q

Table B-5: DFFPC

CLK	CLR	PRE	D	Q
X	H	X	X	L
X	L	H	X	H
L	L	L	X	Q
H	L	L	X	Q
^	L	L	A	A

B.5.6.4 DFFE



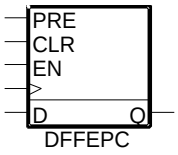
INPUTS: CLK, EN, D

OUTPUTS: Q

Table B-6: DFFE

CLK	EN	D	Q
X	L	X	Q
L	X	X	Q
H	X	X	Q
^	H	A	A

B.5.6.5 DFFEPC



INPUTS: CLK, CLR, PRE, EN, D

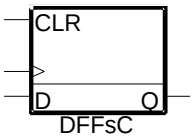
© 2008 QuickLogic Corporation

OUTPUTS: Q

Table B-7: DFFEPC

CLK	CLR	PRE	EN	D	Q
X	H	X	X	X	L
X	L	H	X	X	H
X	L	L	L	X	Q
L	L	L	X	X	Q
H	L	L	X	X	Q
^	L	L	H	A	A

B.5.6.6 DFFsC



INPUTS: CLK, CLR, D

OUTPUTS: Q

Table B-8: DFFsC

CLK	CLR	D	Q
^	H	X	L
L	X	X	Q
H	X	X	Q
^	L	A	A

B.5.6.7 TFF

INPUTS: CLK, T

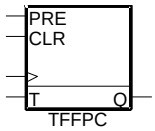
OUTPUTS: Q

Table B-9: TFF

CLK	T	Q
L	X	Q
H	X	Q
^	L	Q
^	H	/Q

© 2008 QuickLogic Corporation

B.5.6.8 TFFPC



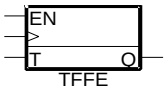
INPUTS: CLK, CLR, PRE, T

OUTPUTS: Q

Table B-10: TFFPC

CLK	CLR	PRE	T	Q
X	H	X	X	L
X	L	H	X	H
L	L	L	X	Q
H	L	L	X	Q
^	L	L	L	Q
^	L	L	H	/Q

B.5.6.9 TFFE



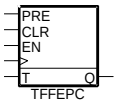
INPUTS: CLK, EN, T

OUTPUTS: Q

Table B-11: TFFE

CLK	EN	T	Q
X	L	X	Q
L	X	X	Q
H	X	X	Q
^	H	L	Q
^	H	H	/Q

B.5.6.10 TFFEPC



inputs: CLK, CLR, PRE, EN, T

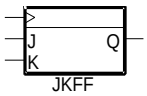
outputs: Q

© 2008 QuickLogic Corporation

Table B-12: TFFEPC

CLK	CLR	PRE	EN	T	Q
X	H	X	X	X	L
X	L	H	X	X	H
X	L	L	L	X	Q
L	L	L	X	X	Q
H	L	L	X	X	Q
^	L	L	H	L	Q
^	L	L	H	H	/Q

B.5.6.11 JKFF



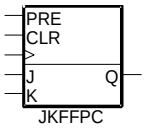
INPUTS: CLK, J, K

OUTPUTS: Q

Table B-13: JKFF

CLK	J	K	Q
L	X	X	Q
H	X	X	Q
^	L	L	Q
^	L	H	L
^	H	L	H
^	H	H	/Q

B.5.6.12 JKFFPC



INPUTS: CLK, CLR, PRE, J, K

OUTPUTS: Q

Table B-14: JKFFPC

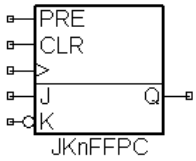
CLK	CLR	PRE	J	K	Q
X	L	H	X	X	H
X	H	X	X	X	L
L	L	L	X	X	Q

© 2008 QuickLogic Corporation

Table B-14: JKFFPC (Continued)

CLK	CLR	PRE	J	K	Q
H	L	L	X	X	Q
^	L	L	L	L	Q
^	L	L	L	H	L
^	L	L	H	L	H
^	L	L	H	H	/Q

B.5.6.13 JKnFFPC

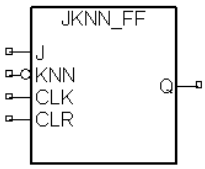


INPUTS: CLK, CLR, PRE, J, /K
OUTPUT: Q

Table B-15: JKnFFPC

CLK	CLR	PRE	J	/K	Q
X	L	H	X	X	H
X	H	X	X	X	L
L	L	L	X	X	Q
H	L	L	X	X	Q
^	L	L	L	H	Q
^	L	L	L	L	L
^	L	L	H	L	/Q
^	L	L	H	H	H

B.5.6.14 JKNN_FF



© 2008 QuickLogic Corporation

INPUTS: CLK, CLR, J, /KNN
OUTPUT: Q

Table B-16: JKNN_FF

CLK	CLR	J	/KNN	Q
X	H	X	X	L
L	L	X	X	Q
H	L	X	X	Q
^	L	L	H	Q
^	L	L	L	L
^	L	H	L	/Q
^	L	H	H	H

© 2008 QuickLogic Corporation

B.6 Latches

The QuickLogic Macro Library includes a number of transparent latches. DLA is a standard transparent latch with no frills. Two of the latch macros, DLAMUX and DLAEMUX, contain a 2-to-1 multiplexer at the input. The features of each latch macro are described in **Table B-17 on page 416**.

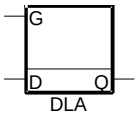
Table B-17: Latch Macro Features

Macro	Description	Clear	Enable
DLA	D-Latch		
DLAC	D-Latch w/ clear	x	
DLAE	D-Latch w/ enable		x
DLAEC	D-Latch w/ enable, clear	x	x
DLAMUX	Latch w/ 2:1 mux input		
DLAEMUX	Latch w/ 2:1 mux input, enable		x
DLAD	Dual Latch w/ common gate		
DLADINV	D-Latch w/ inverted output		
DLAP	D-Latch w/ preset		
DLADE	Dual D Latch w/ common gate, enable		x

NOTE: The last two of the latch macros, DLAD and DLADE, are dual latch macros. Two latches with a common G are implemented in a single logic cell, thereby providing excellent densities.

B.6.1 Macro Descriptions

B.6.1.1 DLA



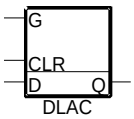
INPUTS: G, D

OUTPUTS: Q

Table B-18: DLA

G	D	Q
L	X	Q
H	A	A

B.6.1.2 DLAC



© 2008 QuickLogic Corporation

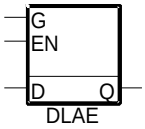
INPUTS: G, CLR, D

OUTPUTS: Q

Table B-19: DLAC

G	CLR	D	Q
X	H	X	L
L	L	X	Q
H	L	A	A

B.6.1.3 DLAE



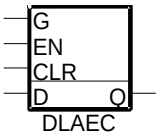
INPUTS: G, EN, D

OUTPUTS: Q

Table B-20: DLAE

G	EN	D	Q
X	L	X	Q
L	H	X	Q
H	H	A	A

B.6.1.4 DLAEC



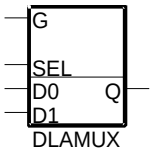
INPUTS: G, CLR, EN, D

OUTPUTS: Q

Table B-21: DLAEC

G	CLR	EN	D	Q
X	H	X	X	L
X	L	L	X	Q
L	L	H	X	Q
H	L	H	A	A

B.6.1.5 DLAMUX



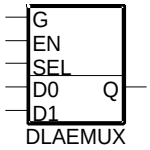
INPUTS: G, SEL, D0, D1

OUTPUTS: Q

Table B-22: DLAMUX

G	SEL	D1	D0	Q
L	X	X	X	Q
H	L	X	A	A
H	H	B	X	B

B.6.1.6 DLAEMUX



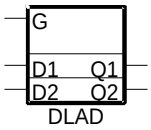
INPUTS: G, EN, SEL, D0, D1

OUTPUTS: Q

Table B-23: DLAEMUX

G	EN	SEL	D1	D0	Q
X	L	X	X	X	Q
L	H	X	X	X	Q
H	H	L	X	A	A
H	H	H	B	X	B

B.6.1.7 DLAD



INPUTS: G, D1, D2

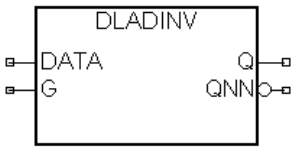
© 2008 QuickLogic Corporation

OUTPUTS: Q1, Q2

Table B-24: DLAD

G	D1	D2	Q1	Q2
L	X	X	Q	Q
H	A	B	A	B

B.6.1.8 DLADINV



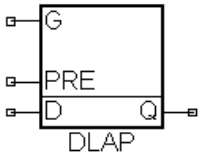
INPUTS: G, DATA

OUTPUTS: Q, QNN

Table B-25: DLADINV

G	DATA	Q	QNN
L	X	Q	QNN
H	A	A	/A

B.6.1.9 DLAP



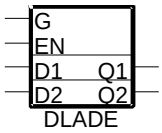
inputs: G, PRE, D

outputs: Q

Table B-26: DLAP

G	PRE	D	Q
X	H	X	H
L	L	X	Q
H	L	A	A

B.6.1.10 DLADE



INPUTS: G, EN, D1, D2

OUTPUTS: Q1, Q2

Table B-27: DLADE

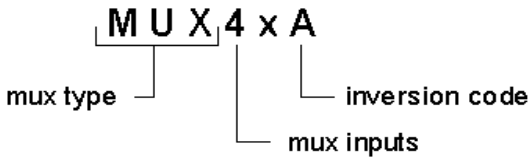
G	EN	D1	D2	Q1	Q2
X	L	X	X	Q	Q
L	H	X	X	Q	Q
H	H	A	B	A	B

© 2008 QuickLogic Corporation

B.7 Multiplexers

The QuickLogic Macro Library includes numerous 4-to-1 and 2-to-1 multiplexers representing all possible inversions at its input. Also included are special flip-flop macros with multiplexers connected to the flip-flop outputs (MUX2FFs).

B.7.1 Naming Conventions



Mux Type—identifies it as a mux.

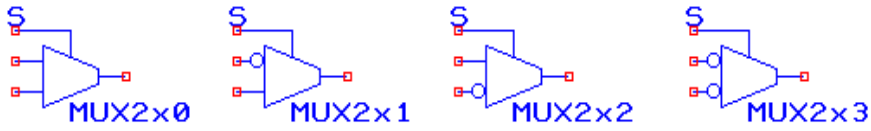
Mux Inputs—specifies the total number of inputs.

X—separates the fields.

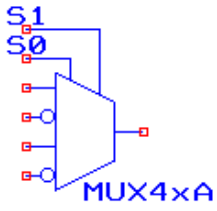
Inversion Code—specifies an inversion code.

The inversion code is used to describe inversions of the data inputs to the mux. The inversion code is a hexadecimal code that goes from zero to three for the 2-input MUXes, and from zero to F for the 4-input MUXes. The coding scheme for the 2-input MUXes is shown in **Figure B-23 on page 421**.

Figure B-23: Coding Scheme for 2-input MUXes



NOTE: Note that the inversion code represents the binary bit pattern of inversions on the data inputs to the mux.

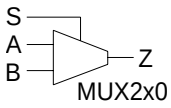


With this understanding we can determine the inversion pattern of a MUX4XA. The inversion code is A which is the hexadecimal equivalent of the binary number 1010 (10 in decimal). The 1's represent inversion bubbles going from bottom to top.

NOTE: The dual 2-to-1 multiplexers (MUX2Dx0, MUX2Dx1, MUX2Dx2, MUX2Dx3) are very efficient macros, as they provide a pair of 2-to-1 multiplexers in a single logic cell.

© 2008 QuickLogic Corporation

B.7.1.1 MUX2x0



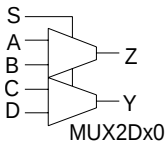
INPUTS: A, B, S

OUTPUTS: Z

Table B-28: MUX2x0

A	B	S	Z
A	B	L	A
A	B	H	B

B.7.1.2 MUX2Dx0



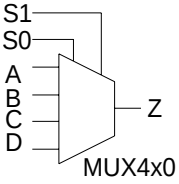
INPUTS: A, B, C, D, S

OUTPUTS: Z, Y

Table B-29: MUX2Dx0

A	B	C	D	S	Z	Y
A	B	C	D	L	A	C
A	B	C	D	H	B	D

B.7.1.3 MUX4x0



INPUTS: A, B, C, D, S1, S0

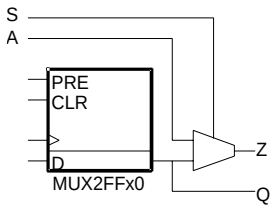
OUTPUTS: Z

Table B-30: MUX4x0

A	B	C	D	S1	S0	Z
A	B	C	D	L	L	A
A	B	C	D	L	H	B
A	B	C	D	H	L	C
A	B	C	D	H	H	D

© 2008 QuickLogic Corporation

B.7.1.4 MUX2FFx0



INPUTS: CLK, CLR, PRE, D, A, S

OUTPUTS: Q, Z

Table B-31: MUX2FFx0

CLK	CLR	PRE	D	A	S	Q	Z
X	H	X	X	X	X	L	X
X	L	H	X	X	X	H	X
^	L	L	B	X	X	B	X
X	X	X	X	A	L	X	A
^	L	L	B	A	H	B	B
^	L	L	B	A	L	B	A

B.8 Registers

NOTE: These registers can only be used in pASIC3, QuickRAM, and QuickPCI designs.

As a design convenience, the QuickLogic Macro Library includes a variety of registers, though they can be built from flip-flops.

Table B-32: Macro Library Registers

Macro	Description	Clear	Enable	Uses DFFs Only
RG4	4-bit Register			x
RG8	8-bit Register			x
RG16	16-bit Register			x
RGC4	4-bit Register w/ clear	x		x
RGC8	8-bit Register w/ clear	x		x
RGC16	16-bit Register w/ clear	x		x
RGE4	4-bit Register w/ enable		x	
RGE8	8-bit Register w/ enable		x	
RGE16	16-bit Register w/ enable		x	
RGEC4	4-bit Register w/ enable, clear	x	x	
RGEC8	8-bit Register w/ enable, clear	x	x	
RGEC16	16-bit Register w/ enable, clear	x	x	

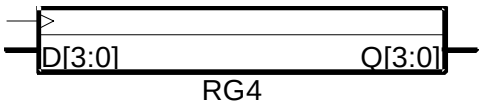
The registers included in the library are a small subset of the possible registers. Other sizes can easily be created by using one of the supplied registers as a starting point.

NOTE: The registers marked in the *Uses DFFs Only* column can be implemented directly in the dedicated flip-flop of the Logic Cell without requiring other logic elements. These registers, therefore, often result in the densest and fastest circuits.

NOTE: The register macros are not available in the PolarPro and PolarPro II device families.

B.8.1 Macro Descriptions

B.8.1.1 RG4



INPUTS: CLK, D[3:0]

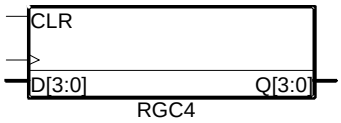
© 2008 QuickLogic Corporation

OUTPUTS: Q[3:0]

Table B-33: RG4

CLK	D[3:0]	Q[3:0]
L	X	Q[3:0]
H	X	Q[3:0]
^	A[3:0]	A[3:0]

B.8.1.2 RGC4



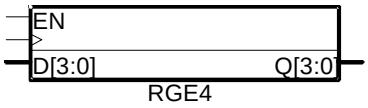
INPUTS: CLK, CLR, D[3:0]

OUTPUTS: Q[3:0]

Table B-34: RGC4

CLK	CLR	D[3:0]	Q[3:0]
X	H	X	0000
L	L	X	Q[3:0]
H	L	X	Q[3:0]
^	L	A[3:0]	A[3:0]

B.8.1.3 RGE4



INPUTS: CLK, EN, D[3:0]

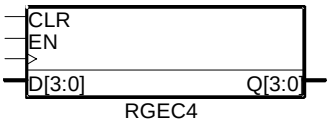
OUTPUTS: Q[3:0]

Table B-35: RGE4

CLK	EN	D[3:0]	Q[3:0]
X	L	X	Q[3:0]
L	H	X	Q[3:0]
H	H	X	Q[3:0]
^	H	A[3:0]	A[3:0]

© 2008 QuickLogic Corporation

B.8.1.4 RGEC4



INPUTS: CLK, CLR, EN, D[3:0]

OUTPUTS: Q[3:0]

Table B-36: RGEC4

CLK	CLR	EN	D[3:0]	Q[3:0]
X	H	X	X	0000
X	L	L	X	Q[3:0]
L	L	H	X	Q[3:0]
H	L	H	X	Q[3:0]
^	L	H	A[3:0]	A[3:0]

© 2008 QuickLogic Corporation

B.9 0.25 um Device Registers

NOTE: These registers can only be used in Eclipse designs.

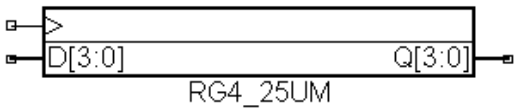
As a design convenience, the QuickLogic Macro Library includes a variety of registers though they can be built from flip-flops.

Table B-37: 0.25 um Device Registers

Macro	Description	Clear	Enable	Use DFFs Only
RG4_25UM	4-bit Register			x
RG8_25UM	8-bit Register			x
RG16_25um	16-bit Register			x
RGC4_25um	4-bit Register w/ clear	x		x
RGC8_25um	8-bit Register w/ clear	x		x
RGC16_25um	16-bit Register w/ clear	x		x
RGE4_25um	4-bit Register w/ enable		x	
RGE8_25um	8-bit Register w/ enable		x	
RGE16_25um	16-bit Register w/ enable		x	
RGEC4_25um	4-bit Register w/ enable, clear		x	
RGEC8_25um	8-bit Register w/ enable, clear		x	
RGEC16_25um	16-bit Register w/ enable, clear		x	
DFF_2	Two D flip-flops			x
DFFC_2	Two D flip-flops w/ clear	x		x
DFFE_2	Two D flip-flops w/ enable		x	
DFFEPC_2	Two D flip-flops w/ preset, enable, clear	x	x	

B.9.1 Macro Descriptions

B.9.1.1 RG4_25UM



INPUTS: CLK, D[3:0]

OUTPUTS: Q[3:0]

Table B-38: RG_25UM

CLK	D[3:0]	Q[3:0]
L	X	Q[3:0]
H	X	Q[3:0]
^	A[3:0]	A[3:0]

B.9.1.2 RGC4_25UM

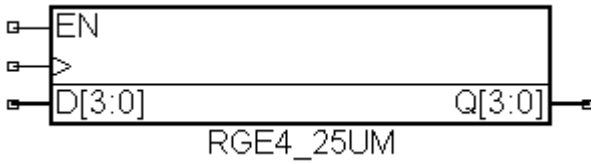
INPUTS: CLK, CLR, D[3:0]

OUTPUTS: Q[3:0]

Table B-39: RGC_25UM

CLK	CLR	D[3:0]	Q[3:0]
X	H	X	0000
L	L	X	Q[3:0]
H	L	X	Q[3:0]
^	L	A[3:0]	A[3:0]

B.9.1.3 RGE4_25UM



INPUTS: CLK, EN, D[3:0]

OUTPUTS: Q[3:0]

Table B-40: RGE4_25UM

CLK	EN	D[3:0]	Q[3:0]
X	L	X	Q[3:0]

© 2008 QuickLogic Corporation

Table B-40: RGE4_25UM

CLK	EN	D[3:0]	Q[3:0]
L	H	X	Q[3:0]
H	H	X	Q[3:0]
^	H	A[3:0]	A[3:0]

B.9.1.4 RGEC4_25UM

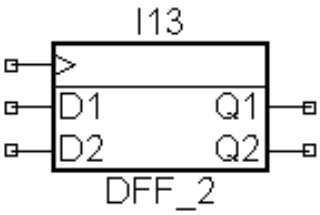
INPUTS: CLK, CLR, EN, D[3:0]

OUTPUTS: Q[3:0]

Table B-41: RGEC4_25UM

CLK	CLR	EN	D[3:0]	Q[3:0]
X	H	X	X	0000
X	L	L	X	Q[3:0]
L	L	H	X	Q[3:0]
H	L	H	X	Q[3:0]
^	L	H	A[3:0]	A[3:0]

B.9.1.5 DFF_2



INPUTS: CLK, CLR, D1, D2[3:0]

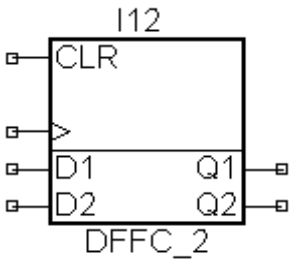
OUTPUTS: Q1, Q2

Table B-42: DFF_2

CLK	D1	D2	Q1	Q2
L	X	X	Q1	Q2
H	X	X	Q1	Q2
^	D1	D2	D1	D2

© 2008 QuickLogic Corporation

B.9.1.6 DFFC_2



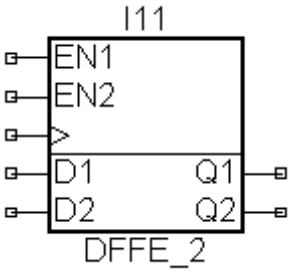
INPUTS: CLK, CLR, D1, D2[3:0]

OUTPUTS: Q1, Q2

Table B-43: DFF_2

CLK	CLR	D1	D2	Q1	Q2
X	H	X	X	L	L
L	L	X	X	Q1	Q2
H	L	X	X	Q1	Q2
^	L	D1	D2	D1	D2

B.9.1.7 DFFE_2



INPUTS: CLK, EN1, EN2, D1, D2

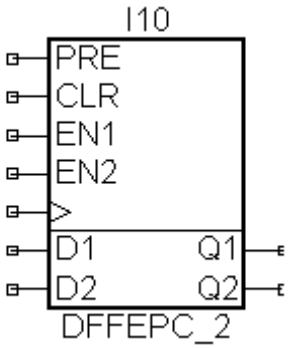
OUTPUTS: Q1, Q2

Table B-44: DFFE_2

CLK	EN1	EN2	D1	D2	Q1	Q2
X	L	L	X	X	Q1	Q2
L	X	X	X	X	Q1	Q2
H	X	X	X	X	Q1	Q2
^	H	H	D1	D2	D1	D2

© 2008 QuickLogic Corporation

B.9.1.8 DFFEPC_2



INPUTS: CLK, CLR, EN1, EN2, PRE, D1, D2

OUTPUTS: Q1, Q2

Table B-45: DFFEPC_2

CLK	CLR	EN1	EN2	PRE	D1	D2	Q1	Q2
X	H	X	X	X	X	X	L	L
X	L	X	X	H	X	X	H	H
X	L	L	L	L	X	X	Q1	Q2
L	L	X	X	L	X	X	Q1	Q2
H	L	X	X	L	X	X	Q1	Q2
^	L	H	H	L	D1	D2	D1	D2

© 2008 QuickLogic Corporation

B.10 Shift Registers

In addition to extremely fast counters, the pASIC architecture can implement extremely fast shift registers. In fact, the shift registers included in the macro library can operate at the maximum logic cell toggle rate.

All of the shift registers offer a common clear control signal.

Table B-46: Macro Library Shift Registers

Macro	Description	Load	Enable	Clear	Direction
SHFT4	4-bit Shift Register w/ load, enable, clear	x	x	x	LSB->MSB
SHFT8	8-bit Shift Register w/ load, enable, clear	x	x	x	LSB->MSB
SHFT16	16-bit Shift Register w/ load, enable, clear	x	x	x	LSB->MSB
BSHFT4	4-bit Bi-Directional Shift Register w/ load, enable, clear	x	x	x	Both
BSHFT8	8-bit Bi-Directional Shift Register w/ load, enable, clear	x	x	x	Both
BSHFT16	16-bit Bi-Directional Shift Register w/ load, enable, clear	x	x	x	Both
LSHFT2Q2	2-bit Dual Shift Register w/ clear			x	LSB->MSB

The unidirectional shifters have a common `LOAD` control signal, which causes the register to be loaded synchronously from the `D` inputs when asserted. The bi-directional shifters have common `S0` and `S1` control signals, which specify the function code.

Table B-47: Bi-Directional Shift Registers

S1	S0	Function
0	0	Hold[a] Value
0	1	Shift Right[b] (LSB-MSB)
1	0	Shift Left[c] (MSB-LSB)
1	1	Load[d] New Value

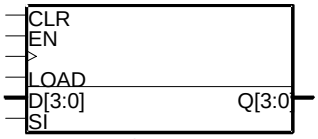
a. The hold function performs no action—the state of the register is unchanged
b. The right shift function loads each bit from its RSI input
c. The left shift function loads each bit from its LSI input
d. The load function performs a parallel load from the D inputs.

LSHFT2Q2 is a dual shift register implemented in one pASIC3, QuickRAM, QuickPCI, or Eclipse Logic Cell by wiring up the logic before the flip-flop to act as a master-slave latch.

© 2008 QuickLogic Corporation

B.10.1 Macro Descriptions

B.10.1.1 SHFT4



INPUTS: CLK, CLR, EN, LOAD, D[3:0], SI

OUTPUTS: Q[3:0]

Table B-48: SHFT4

CLK	CLR	EN	LOAD	D[3:0]	SI	Q[3:0]	
X	H	X	X	X	X	0000	CLEAR
X	L	L	L	X	X	Q[3:0]	HOLD
^	L	X	H	A[3:0]	X	A[3:0]	LOAD
^	L	H	L	X	E	Q[2:0], E	SHIFT

B.10.1.2 BSHFT4

INPUTS: CLK, CLR, S0, S1, D[3:0], RSI, LSI

OUTPUTS: Q[3:0]

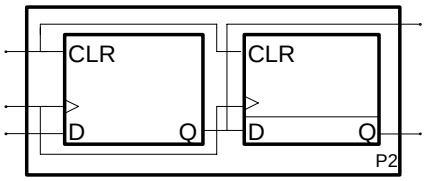
Table B-49: BSHFT4

CLK	CLR	S0	S1	D[3:0]	RSI	LSI	Q[3:0]	
X	H	X	X	X	X	X	0000	CLEAR
X	L	L	L	X	X	X	Q[3:0]	HOLD
^	L	H	H	A[3:0]	X	X	A[3:0]	LOAD
^	L	H	L	X	A	X	A,Q[3:1]	SHFT RHT
^	L	L	H	X	X	A	Q[2:0],A	SHFT LFT

B.10.1.3 LSHFT2Q2

NOTE: This macro can only be used in pASIC3, QuickRAM, QuickPCI, and Eclipse designs.

© 2008 QuickLogic Corporation



LSHFT2Q2

INPUTS: CLK, CLR, D

OUTPUTS: Q0, Q1

Table B-50: LSHFT2Q2

CLK	CLR	D	Q0	Q1
X	H	X	0	0
first ^	L	A1	A1	0
second ^	L	A2	A2	A1

NOTE: When using the LSHFT2Q2, you must use the internal global clock network for the clock input. The GCLKBUFF macro (or GCLKBUFF_25um for Eclipse devices) is used to specify an internally generated global clock net. This macro is found in the OTHER symbol library.

© 2008 QuickLogic Corporation

B.11 Up Counters

The QuickLogic Macro Library has a variety of up counter macros. There are ripple-carry counters and parallel-carry counters. An E indicates that an enable is available; an L indicates a loadable capability.

Table B-51: Up Counters Macros

Macro	Description	Load	Enable	Logic Cells in Crit-Path	# Logic Cells
RCNT4	4-Bit Ripple Up-Counter w/ RCO	x	x	4	4
RCNT8	8-Bit Ripple Up-Counter w/ RCO	x	x	8	8
RCNT16	16-Bit Ripple Up-Counter w/ RCO	x	x	16	16
UCNTE4	4-Bit Fast Up-Counter w/ enable		x	1	4
UCNTE8	8-Bit Fast Up-Counter w/ enable		x	1	11
UCNTE16	16-Bit Fast Up-Counter w/ enable		x	1	24
UCNTE24	24-Bit Fast Up-Counter w/ enable		x	1	38
UCNTE32	32-Bit Fast Up-Counter w/ enable		x	1	52
UCNTL4	4-Bit Fast Up-Counter w/ load	x		1	4
UCNTL8	8-Bit Fast Up-Counter w/ load	x		1	11
UCNTL16	16-Bit Fast Up-Counter w/ load	x		1	24
UCNTL24	24-Bit Fast Up-Counter w/ load	x		1	38
UCNTL32	32-Bit Fast Up-Counter w/ load	x		1	52
UCNTX4	4-Bit Fast Up-Counter w/ load, enable	x	x	1.5	4
UCNTX8	8-Bit Fast Up-Counter w/ load, enable	x	x	1.5	11
UCNTX16	16-Bit Fast Up-Counter w/ load, enable	x	x	1.5	24
UCNTX24	24-Bit Fast Up-Counter w/ load, enable	x	x	1.5	36
UCNTX32	32-Bit Fast Up-Counter w/ load, enable	x	x	1.5	50
UPFLCT4	4-bit Fast Up-Counter w/ load (Reversed Bits)	x		1	4
UPFLCT8	8-bit Fast Up-Counter w/ load (Reversed Bits)	x		1	11
UPFLCT16	16-bit Fast Up-Counter w/ load (Reversed Bits)	x		1	21
UPFLCT24	24-bit Fast Up- Counter w/ load (Reversed Bits)	x		1	33
UPFLCT32	32-bit Fast Up-Counter w/ load (Reversed Bits)	x		1	45
UPFXCT4	4-bit Fast Up-Counter w/ load, enable (Reversed Bits)	x	x	1.5	4
UPFXCT8	8-bit Fast Up-Counter w/ load, enable (Reversed Bits)	x	x	1.5	9

© 2008 QuickLogic Corporation

Table B-51: Up Counters Macros (Continued)

Macro	Description	Load	Enable	Logic Cells in Crit-Path	# Logic Cells
UPFXCT16	16-bit Fast Up-Counter w/ load, enable (Reversed Bits)	x	x	1.5	19
UPFXCT24	24-bit Fast Up-Counter w/ load, enable (Reversed Bits)	x	x	1.5	31
UPFXCT32	32-bit Fast Up-Counter w/ load, enable (Reversed Bits)	x	x	1.5	43

The UCNT... counters are performance optimized; they have been tuned to reduce internal fanout as much as possible. Astute observers will recognize that these counters do not include a ripple carry output. Their pre-scaled design (based loosely on the 'LS163) does not produce a true RCO signal at the output of the last stage. If a very fast RCO is required, directly gate the Q[X:0] output bus. These counters are for all device families.

Each of these macros has an asynchronous clear input.

NOTE: Fast counters with load or enable inputs (UCNTL.. or UCNTX..) operate very fast when counting only (> 70 MHz), but if they are loaded or disabled, they need to be clocked at a slower rate (> 30 MHz). Some designs use a fast clock, but allow multiple clocks for a load.

NOTE: All of the fast up counters (UCNT...) above 4 bits use replicated internal flip-flops for the lowest bits of the counter. Because of this, it is important to clear or load the counters before counting to initialize these replicated bits.

NOTE: The enable-only and load-only (UCNTE, UCNTL, DCNTE, DCNTL) counters are constructed with only one level of logic cells between flip-flops. These counters offer slightly better performance than the enable-and-load counters (UCNTX, DCNTX).

NOTE: Only UCNTE4, UCNTL4, UCNTX4, UPFLCT4, and UPFXCT4 up counter macros are available in the PolarPro and PolarPro II device families.

B.11.1 QuickLogic Device-Specific Counters

NOTE: These macros can only be used in pASIC3, QuickRAM, QuickPCI, and Eclipse devices.

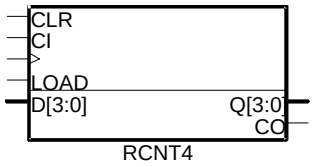
Table B-52: UCT Counters

Macro	Description	Load	Enable	Logic Cells in Crit-Path	# Logic Cells
UCT8P2	8-Bit, Fast Counter w/ reset			1	9
UCTE16P2	16-Bit, Fast Counter w/ reset, enable		x	1	18
UCTX16P2	16-Bit, Fast Counter w/ reset, enable, load	x	x	1.5	22

NOTE: The UCT counters are new high-speed counter macros. Change the second sentence to: These macros will only work for the pASIC3, QuickRAM, QuickPCI, and Eclipse device families. Unlike the pre-scaled UCT and UDCNT counters, these counters have optimized enable and load paths to offer true 100–200 MHz performance, even with load and enable control. Speed depends on device, speed grade, operating conditions and placement. These macros are also available in HDL format.

© 2008 QuickLogic Corporation

B.11.1.1 RCNT4



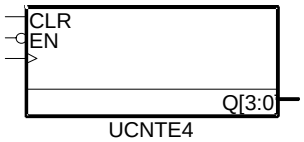
INPUTS: CLK, CLR, LOAD, CI, D[3:0]

OUTPUTS: Q[3:0], CO

Table B-53: RCNT4

CLK	CLR	LOAD	CI	D[3:0]	Q[3:0]-old	Q[3:0]-new	CO
X	H	X	X	X	XXXX	0000	X
^	L	H	X	A[3:0]	XXXX	A[3:0]	X
^	L	L	L	X	Q[3:0]	Q[3:0]	L
^	L	L	H	X	Q[3:0]	Q[3:0]+1	L
X	L	X	H	X	1111	1111	H

B.11.1.2 UCNTE4



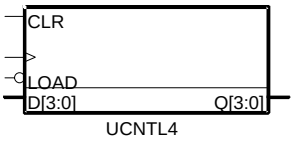
INPUTS: CLK, CLR, EN

OUTPUTS: Q[3:0]

Table B-54: UCNTE4

CLK	CLR	EN	Q[3:0]-old	Q[3:0]-new
X	H	X	XXXX	0000
X	L	H	Q[3:0]	Q[3:0]
^	L	L	Q[3:0]	Q[3:0]+1

B.11.1.3 UCNTL4



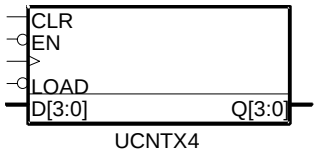
INPUTS: CLK, CLR, LOAD, D[3:0]

OUTPUTS: Q[3:0]

Table B-55: UCNTL4

CLK	CLR	LOAD	D[3:0]	Q[3:0]-old	Q[3:0]-new
X	H	X	X	XXXX	0000
^	L	L	A[3:0]	XXXX	A[3:0]
^	L	H	X	Q[3:0]	Q[3:0]+1

B.11.1.4 UCNTX4



INPUTS: CLK, CLR, EN, LOAD, D[3:0]

OUTPUTS: Q[3:0]

Table B-56: UCNTX4

CLK	CLR	EN	LOAD	D[3:0]	Q[3:0]-old	Q[3:0]-new
X	H	X	X	X	XXXX	0000
^	L	H	H	X	Q[3:0]	Q[3:0]
^	L	X	L	A[3:0]	XXXX	A[3:0]
^	L	L	H	X	Q[3:0]	Q[3:0]+1

B.11.1.5 UCT8P2

INPUTS: CLR, CLK

OUTPUTS: Q[7:0]

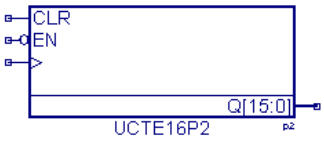
Table B-57: UCT8P2

CLK	CLR	Q[7:0] old-binary	Q[7:0] new-binary
X	H	Q[7:0]	000 000 00
X	L	Q[7:0]	Q[7:0]
^	L	Q[7:0]	Q[7:0]+1

NOTE: Fast 8-bit Binary Up Counter. Reset required for proper operation.

© 2008 QuickLogic Corporation

B.11.1.6 UCTE16P2



INPUTS: CLR, EN, CLK

OUTPUTS: Q[15:0]

Table B-58: UCTE16P2

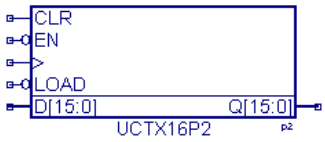
CLK	CLR	EN	Q[15:0]-old	Q[15:0]-new
X	H	X	Q[15:0]	0000 0000 0000 0000
X	L	H	Q[15:0]	Q[15:0]
^	L	L	Q[15:0]	Q[15:0]+1

NOTE: Fast 16-bit Binary Up Counter.

The pins description is as follows:

- Asynchronous Reset (active high)
- Synchronous Enable (active low)
- Reset required for proper operation

B.11.1.7 UCTX16P2



INPUTS: CLR, EN, CLK, LOAD, D[15:0]

OUTPUTS: Q[15:0]

Table B-59: UCTX16P2

CLK	CLR	EN	LOAD	D[15:0]	Q[15:0]-old	Q[15:0]-new
X	H	X	X	X	Q[15:0]	0000 0000 0000 0000
^	L	H	H	X	Q[15:0]	Q[15:0]
^	L	X	L	A[15:0]	Q[15:0]	A[15:0]
^	L	L	H	X	Q[15:0]	Q[15:0]+1

NOTE: Fast 16-bit Loadable Binary Up Counter.

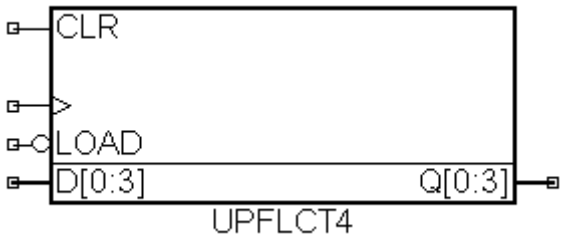
The pins description is as follows:

- Asynchronous Reset (active high)

© 2008 QuickLogic Corporation

- Synchronous Enable (active low)
- Load or Reset required for proper operation

B.11.1.8 UPFLCT4



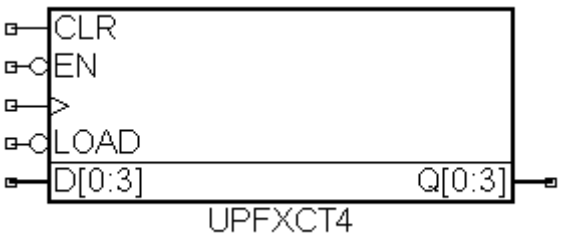
inputs: CLK, CLR, LOAD, D[0:3]

outputs: Q[0:3]

Table B-60: UPFLCT4

CLK	CLR	LOAD	D[0:3]	Q[0:3]-old	Q[0:3]-new
X	H	X	X	XXXX	0000
^	L	L	A[0:3]	XXXX	A[0:3]
^	L	H	X	Q[0:3]	Q[0:3]+1

B.11.1.9 UPFXCT4



INPUTS: CLK, CLR, EN, LOAD, D[0:3]

OUTPUTS: Q[0:3]

Table B-61: UPFXCT4

CLK	CLR	EN	LOAD	D[0:3]	Q[0:3]-old	Q[0:3]-new
X	H	X	X	X	XXXX	0000
^	L	H	H	X	Q[0:3]	Q[0:3]
^	L	X	L	A[0:3]	XXXX	A[0:3]
^	L	L	H	X	Q[0:3]	Q[0:3]+1

© 2008 QuickLogic Corporation

B.12 Down Counters

The QuickLogic Macro Library has a variety of down counter macros. There are ripple-carry counters and parallel-carry counters. An E indicates that an enable is available; an L indicates a loadable capability.

Table B-62: Down Counters

Macro	Description	Load	Enable	Logic Cells in Crit-Path	# Logic Cells
DCNTE4	4-Bit Fast Down-Counter w/ enable		x	1	4
DCNTE8	8-Bit Fast Down-Counter w/ enable		x	1	11
DCNTE16	16-Bit Fast Down-Counter w/ enable		x	1	23
DCNTE24	24-Bit Fast Down-Counter w/ enable		x	1	38
DCNTE32	32-Bit Fast Down-Counter w/ enable		x	1	51
DCNTL4	4-Bit Fast Down-Counter w/ load	x		1	4
DCNTL8	8-Bit Fast Down-Counter w/ load	x		1	11
DCNTL16	16-Bit Fast Down-Counter w/ load	x		1	23
DCNTL24	24-Bit Fast Down-Counter w/ load	x		1	38
DCNTL32	32-Bit Fast Down-Counter w/ load	x		1	51
DCNTX4	4-Bit Fast Down-Counter w/ load, enable	x	x	1.5	4
DCNTX8	8-Bit Fast Down-Counter w/ load, enable	x	x	1.5	11
DCNTX16	16-Bit Fast Down-Counter w/ load, enable	x	x	1.5	22
DCNTX24	24-Bit Fast Down-Counter w/ load, enable	x	x	1.5	36
DCNTX32	32-Bit Fast Down-Counter w/ load, enable	x	x	1.5	50

The DCNT... counters are performance optimized; they have been tuned to reduce internal fanout as much as possible. Astute observers will recognize that these counters do not include a ripple carry output. Their pre-scaled design (based loosely on the TTL163) does not produce a true RCO signal at the output of the last stage. If a very fast RCO is required, directly gate the Q[X:0] output bus.

Each of these macros has an asynchronous clear input.

NOTE: Counters with load or enable inputs (DCNTL.. or DCNTX..) operate very fast when counting only (> 70 MHz), but if they are loaded or disabled, they need to be clocked at a slower rate (> 30 MHz). Some designs use a fast clock, but allow multiple clocks for a load.

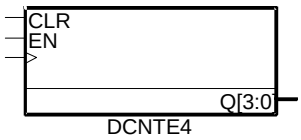
NOTE: All of the down counters above four bits use replicated internal flip-flops for the lowest bits of the counter. Because of this, it is important to clear or load the counters before counting, because these replicated bits must be initialized to the same value.

NOTE: Only DCNTE4, DCNTL4 and DCNTX4 down counter macros are available in the PolarPro and PolarPro II device families.

© 2008 QuickLogic Corporation

B.12.1 Macro Descriptions

B.12.1.1 DCNTE4



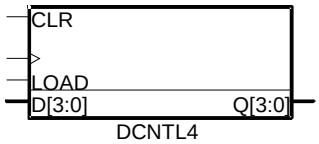
INPUTS: CLK, CLR, EN

OUTPUTS: Q[3:0]

Table B-63: DCNTE4

CLK	CLR	EN	Q[3:0]-old	Q[3:0]-new
X	H	X	XXXX	0000
X	L	L	Q[3:0]	Q[3:0]
^	L	H	Q[3:0]	Q[3:0]-1

B.12.1.2 DCNTL4



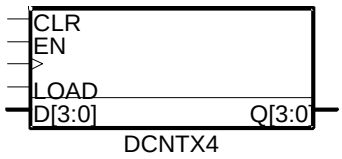
INPUTS: CLK, CLR, LOAD, D[3:0]

OUTPUTS: Q[3:0]

Table B-64: DCNTL4

CLK	CLR	LOAD	D[3:0]	Q[3:0]-old	Q[3:0]-new
X	H	X	X	XXXX	0000
^	L	H	A[3:0]	XXXX	A[3:0]
^	L	L	X	Q[3:0]	Q[3:0]-1

B.12.1.3 DCNTX4



INPUTS: CLK, CLR, EN, LOAD, D[3:0]

© 2008 QuickLogic Corporation

OUTPUTS: Q[3:0]

Table B-65: DCNTX4

CLK	CLR	EN	LOAD	D[3:0]	Q[3:0]	Q[3:0]
X	H	X	X	X	XXXX	0000
^	L	L	L	X	Q[3:0]	Q[3:0]
^	L	X	H	A[3:0]	XXXX	A[3:0]
^	L	H	L	X	Q[3:0]	Q[3:0]-1

© 2008 QuickLogic Corporation

B.13 Up/Down Counters

The QuickLogic Macro Library has an up/down counter macro with various bits. It can be enabled both as an active high or low.

Table B-66: Up/Down Counters

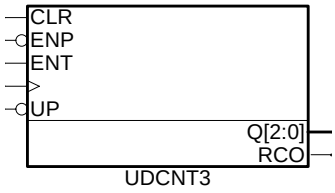
Macro	Description	Enable	Logic Cells in Crit-Path	# Logic Cells
UDCNT3	3-Bit Up/Down-Counter w/ enable, rco	x	1	4
UDCNT6	6-Bit Up/Down-Counter w/ enable, rco	x	2	8
UDCNT12	12-Bit Up/Down-Counter w/ enable, rco	x	2	16
UDLF6	6-Bit, Fast Up/Down-Counter			16
UDLF12	12-Bit, Fast Up/Down-Counter			27
UDLF18	18-Bit, Fast Up/Down-Counter			52
UDLF24	24-Bit, Fast Up/Down-Counter			70

NOTE: UDLF6, UDLF12, UDLF18, and UDLF24 are pASIC3, QuickRAM, QuickPCI, and Eclipse-specific up/down counters.

NOTE: Only the UDCNT3 up/down counter macro is available in the PolarPro and PolarPro II device families.

B.13.1 Macro Descriptions

B.13.1.1 UDCNT3



INPUTS: CLK, CLR, ENP, ENT, UP

OUTPUTS: Q[2:0], RCO

Table B-67: UDCNT3

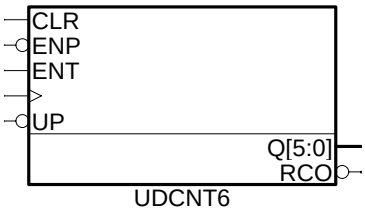
CLK	CLR	ENP	ENT	UP	Q[2:0]-old (binary)	Q[2:0]-new (binary)	RCO
X	H	X	X	X	XXX	000	X
X	L	H	X	X	Q[2:0]	Q[2:0]	X
X	L	X	L	X	Q[2:0]	Q[2:0]	L
X	L	X	H	L	HHH	XXX	H

© 2008 QuickLogic Corporation

Table B-67: UDCNT3

CLK	CLR	ENP	ENT	UP	Q[2:0]-old (binary)	Q[2:0]-new (binary)	RCO
X	L	X	H	H	LLL	XXX	H
^	L	L	H	L	Q[2:0]	Q[2:0]+1	X
^	L	L	H	H	Q[2:0]	Q[2:0]-1	X

B.13.1.2 UDCNT6



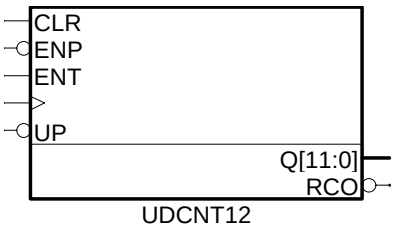
INPUTS: CLK, CLR, ENP, ENT, UP

OUTPUTS: Q[5:0], RCO

Table B-68: UDCNT6

CLK	CLR	ENP	ENT	UP	Q[5:0]-old (binary)	Q[5:0]-new (binary)	RCO
X	H	X	X	X	XXXXXX	000000	X
X	L	H	X	X	Q[5:0]	Q[5:0]	X
X	L	X	L	X	Q[5:0]	Q[5:0]	H
X	L	L	H	L	HHHHHH	XXXXXX	L
X	L	L	H	H	LLLLLL	XXXXXX	L
^	L	L	H	L	Q[5:0]	Q[5:0]+1	X
^	L	L	H	H	Q[5:0]	Q[5:0]-1	X

B.13.1.3 UDCNT12



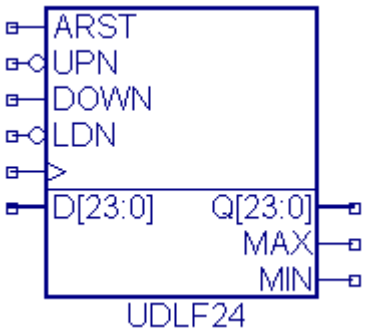
INPUTS: CLK, CLR, ENP, ENT, UP

OUTPUTS: Q[11:0], RCO

Table B-69: UDCNT12

CLK	CLR	ENP	ENT	UP	Q[11:0]-old (hexadecimal)	Q[11:0]-new (hexadecimal)	RCO
X	H	X	X	X	XXX	000	X
X	L	H	X	X	Q[11:0]	Q[11:0]	X
X	L	X	L	X	Q[11:0]	Q[11:0]	H
X	L	X	H	L	FFF	XXX	L
X	L	X	H	H	000	XXX	L
^	L	L	H	L	Q[11:0]	Q[11:0]+1	X
^	L	L	H	H	Q[11:0]	Q[11:0]-1	X

B.13.1.4 UDLF24



INPUTS: ARST, CLK, UPN, LDN, DOWN, D[23:0]

OUTPUTS: Q[23:0], MAX, MIN

Table B-70: UDLF24

ARST	CLK	UPN	LDN	DOWN	D[23:0]	Q[23:0]-old	Q[23:0]-new	max	min
H	X	X	X	X	0	0	0	L	H
L	^	L	L	L	D[23:0]	D[23:0]	D[23:0]+1	a	a
L	^	H	L	H	D[23:0]	D[23:0]	D[23:0]-1	a	a
L	^	L	L	H	D[23:0]	D[23:0]	Q[23:0]	a	a

a. Min output high when all bits of Q[23:0] go low; Max output high when all bits of Q[23:0] go high.

NOTE: UDLF6, UDLF12, UDFL18, and UDLF24 are fast, loadable, up/down binary counter macros for pASIC3, QuickRAM, QuickPCI, and Eclipse devices.

The pins description is as follows:

- Asynchronous Reset (active high)
- Synchronous Load (active low)
- Synchronous Up (active low)

© 2008 QuickLogic Corporation

- Synchronous Down (active high)
- MIN Output when all bits are low
- MAX Output when all bits are high
- Disabled when Up and Down are both active/inactive
- Initial Load/Reset required for proper operation

B.14 pASIC Arithmetic

The pASIC logic cell is well suited for high-speed arithmetic. The conditional-sum macros (`ADDxx`, `SUBxx`) offer exceptional speed with excellent density.

Table B-71: pASIC Logic Cell Tables

Macro	Description	Logic Cells in Crit-Path	# Logic Cells
HADD1	Half-Adder	1	1
FADD1	Full-Adder	1	2
RADD4	4-bit Ripple Adder with CI,CO	4	8
RADD8	8-bit Ripple Adder with CI,CO	8	16
RADD16	16-bit Ripple Adder with CI,CO	16	32
ADD4	4-bit Fast Adder	2	7
ADD8	8-bit Fast Adder	3	20
ADD16	16-bit Fast Adder	4	48
ADD32	32-bit Fast Adder	5.5	120
FSTADD4	pASIC3, QuickRAM, QuickPCI, Eclipse-specific 4-bit Fast Adder	2	7
FADD1_P2	pASIC3, QuickRAM, QuickPCI, Eclipse-specific Full Adder	1	1
FSTADD8	ppASIC3, QuickRAM, QuickPCI, Eclipse-specific 8-bit Fast Adder	3	19
FSTADD16	pASIC3, QuickRAM, QuickPCI, Eclipse-specific 16-bit Fast Adder	3	52
SUB4	4-bit Fast Subtractor (2's complement)	2	7
SUB8	8-bit Fast Subtractor (2's complement)	3	20
SUB16	16-bit Fast Subtractor (2's complement)	4	48
SUB32	32-bit Fast Subtractor (2's complement)	5.5	120
ACCUM4	4-bit Fast Accumulator	2	7
ACCUM8	8-bit Fast Accumulator	3	20
ACCUM16	16-bit Fast Accumulator	4	48
ACCUM32	32-bit Fast Accumulator	5.5	120
MULT4X4	4-by-4 Multiplier	5	30
COMP4	4-bit Equality Comparator	2	3
COMP8	8-bit Equality Comparator	2	5

© 2008 QuickLogic Corporation

Table B-71: pASIC Logic Cell Tables (Continued)

Macro	Description	Logic Cells in Crit-Path	# Logic Cells
COMP12	12-bit Equality Comparator	2	7
ECOMP2	pASIC3, QuickRAM, QuickPCI, Eclipse, and EclipsePlus-specific 2-bit Equality Comparator	1	1
ECOMP4	pASIC3, QuickRAM, QuickPCI, Eclipse, and EclipsePlus-specific 4-bit Equality Comparator	2	3
ECOMP8	pASIC3, QuickRAM, QuickPCI, Eclipse, and EclipsePlus-specific 8-bit Equality Comparator	2	5
ECOMP16	pASIC3, QuickRAM, QuickPCI, Eclipse, and EclipsePlus-specific 16-bit Equality Comparator	2	9
ECOMP32	pASIC3, QuickRAM, QuickPCI, Eclipse, and EclipsePlus-specific 32-bit Equality Comparator	2	17

NOTE: In **Table B-71** all arithmetic macros except FSTADD4/8/16 and ECOMP2/4/8/16/32 are available in the PolarPro and PolarPro II device family.

B.14.1 Macro Descriptions

B.14.1.1 HADD1

INPUTS: A, B

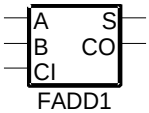
OUTPUTS: S, CO

Table B-72: HADD1

A	B	S	CO
L	L	L	L
L	H	H	L
H	L	H	L
H	H	L	H

HADD1 is a simple half-adder macro which can be implemented in less than one logic cell. The outputs of the half-adder are S and CO, the sum and carry-out respectively.

B.14.1.2 FADD1



INPUTS: A, B, CI

OUTPUTS: S, CO

© 2008 QuickLogic Corporation

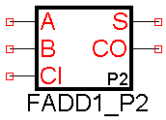
Table B-73: FADD1

A	B	CI	S	CO
L	L	L	L	L
L	L	H	H	L
L	H	L	H	L
L	H	H	L	H
H	L	L	H	L
H	L	H	L	H
H	H	L	L	H
H	H	H	H	H

`FADD1` is a full-adder which requires two logic cells. This soft macro contains one `XOR3i0` and one `MAJ3i0` macro. As noted in the hard macro section earlier, both of these macros have efficient implementation, which require less than an entire logic cell. This property is fully utilized in the multiplier macro presented shortly.

B.14.1.3 FADD1_P2

NOTE: This macro can only be used in pASIC3, QuickRAM, QuickPCI, and Eclipse designs.



INPUTS: A, B, CI

OUTPUTS: S, CO

Table B-74: FADD1_P2

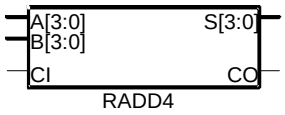
A	B	CI	S	CO
L	L	L	L	L
L	L	H	H	L
L	H	L	H	L
L	H	H	L	H
H	L	L	H	L
H	L	H	L	H
H	H	L	L	H
H	H	H	H	H

FADD1_P2 is a full-adder which requires only one pASIC3, QuickRAM, QuickPCI, or Eclipse logic cell. Both the sum and carry out are generated with the combinatorial logic. In addition, the sum can also be fed into the flip-flop of the same logic cell (enabling the sum to be

© 2008 QuickLogic Corporation

registered in synchronous designs). This soft macro is optimized for the lowest internal fanout, and the highest speeds. Use this adder for pASIC3, QuickRAM, QuickPCI, and Eclipse designs to help reduce the logic cell utilization, and take advantage of the wide fanin of the pASIC3, QuickRAM, QuickPCI, or Eclipse logic cell.

B.14.1.4 RADD4



INPUTS: A[3:0], B[3:0], CI

OUTPUTS: S[3:0], CO

Table B-75: RADD4

A[3:0]	B[3:0]	CI	S[3:0]	CO	if
A	B	CI	A+B+CI	H	A+B+CI>1111
A	B	CI	A+B+CI	L	A+B+CI<=1111

RADD4, RADD8, and RADD16 are ripple-carry adders of 4, 8, and 16 bits respectively. Each utilizes the FADD1 full-adder macro in a ripple-carry arrangement. This provides excellent density at the cost of additional logic delays, compared to the conditional-sum adders to be introduced shortly.

B.14.1.5 ADD4

INPUTS: A[3:0], B[3:0]

OUTPUTS: Q[3:0]

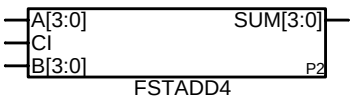
Table B-76: ADD4

A[3:0]'	B[3:0]	Q[3:0]
A	B	A + B

The ADD macros are conditional-sum adders, which have been tuned for exceptionally low internal fanouts and exceptionally high speeds.

B.14.1.6 FSTADD4

NOTE: This macro can only be used in pASIC3, QuickRAM, QuickPCI, and Eclipse designs.



INPUTS: A[3:0], B[3:0], C1

© 2008 QuickLogic Corporation

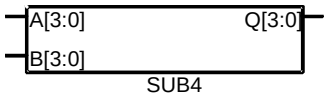
OUTPUTS: SUM Q[3:0]

Table B-77: FSTADD4

A[3:0]	B[3:0]	C1	SUM[3:0]
A	B	C1	A + B + C1

The FSTADD macros are propagate-generate adders, which have been tuned for exceptionally high speeds.

B.14.1.7 SUB4



INPUTS: A[3:0], B[3:0]

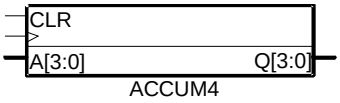
OUTPUTS: Q[3:0]

Table B-78: SUB4

A[3:0]	B[3:0]	Q[3:0]
A	B	A - B

The SUB macros are conditional-sum adders, which have been tuned for exceptionally low internal fanouts and exceptionally high speeds.

B.14.1.8 ACCUM4



INPUTS: CLK, CLR, A[3:0]

OUTPUTS: Q[3:0]

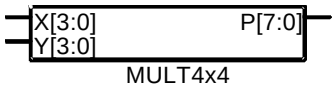
Table B-79: ACCUM4

CLK	CLR	A[3:0]	Q[3:0]-old	Q[3:0]-new
X	H	XXXX	XXXX	0000
^	L	A	Q	Q + A

The ACCUM macros are conditional-sum adders, which have been tuned for exceptionally low internal fanouts and exceptionally high speeds.

© 2008 QuickLogic Corporation

B.14.1.9 MULT4x4



INPUTS: X[3:0], Y[3:0]

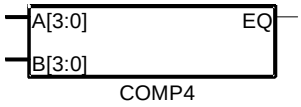
OUTPUTS: P[7:0]

Table B-80: MULT4x4

X[3:0]	Y[3:0]	P[7:0]
A	B	A times B

MULT4x4 is a four-by-four multiplier, which offers excellent performance. This macro's design takes advantage of the XOR utilization property—two AND gates and a 3-input XOR gate can be packed into a single pASIC logic cell.

B.14.1.10 COMP4



INPUTS: A[3:0], B[3:0]

OUTPUTS: EQ

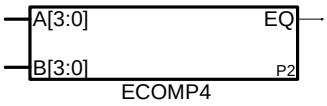
Table B-81: COMP4

A[3:0]	B[3:0]	EQ	if
A	B	L	A = B
A	B	H	A = B

These macros (COMP4, COMP8, COMP12) require two logic levels and features excellent performance. Larger equality comparators can be constructed easily by ANDing two or more of these macros.

B.14.1.11 ECOMP4

NOTE: This macro can only be used in pASIC3, QuickRAM, QuickPCI, and Eclipse designs



INPUTS: A[3:0], B[3:0]

OUTPUTS: EQ

© 2008 QuickLogic Corporation

Table B-82: ECOMP4

A[3:0]	B[3:0]	EQ	if
A	B	L	A != B
A	B	H	A = B

These macros (ECOMP2, ECOMP4, ECOMP8, ECOMP16, ECOMP32) require two logic levels and feature excellent performance. Larger equality comparators can be constructed easily by AND-ing two or more of these macros.

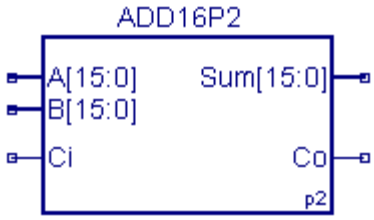
B.14.2 QuickLogic Device-Specific Adders and Subtracters

NOTE: These macros can only be used in pASIC3, QuickRAM, QuickPCI, and Eclipse devices.

Table B-83: Adders and Substracters Macros

Macro	Description	Logic Cells in Crit-Path	# Logic Cells
ADD16P2	16-bit carry select Adder with CI, CO	4	53
ADD32P2	32-bit carry select Adder with CI, CO	5	120
SUB16P2	16-bit borrow select Adder with BI, BO	4	53
SUB32P2	32-bit borrow select Adder with BI, BO	5	120

B.14.2.1 ADD16P2



INPUTS: A[15:0], B[15:0], CI

OUTPUTS: SUM[15:0], CO

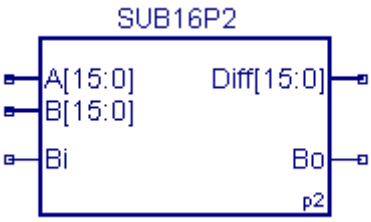
Table B-84: ADD16P2

A[15:0]	B[15:0]	CI	SUM[15:0]	CO	if
A	B	CI	A+B+CI	H	A+B+CI> 1111 1111 1111 1111
A	B	CI	A+B+CI	L	A+B+CI< 1111 1111 1111 1111

ADD16P2 and ADD32P2 are fast adders with carry select for the pASIC3, QuickRAM, QuickPCI, and Eclipse device families.

© 2008 QuickLogic Corporation

B.14.2.2 SUB16P2



INPUTS: A[15:0], B[15:0], BI

OUTPUTS: DIFF[15:0], BO

Table B-85: SUB16P2

A[15:0]	B[15:0]	BI	DIFF[15:0]	BO	if
A	B	BI	A-B-BI	0	A>(B + B1)
A	B	BI	A-B-BI	1	A<(B + B1)

SUB16P2 and SUB32P2 are fast Subtracters with borrow select for the pASIC3, QuickRAM, QuickPCI, and Eclipse device families.

© 2008 QuickLogic Corporation

B.15 Combinatorial Macros

The QuickLogic Macro Library includes two, three, four, and five input XOR (exclusive-or) and XNOR (exclusive-nor, also known as equivalence) gates. Their names use the same terminology as the AND, NOR, NAND, OR gates, although input inversion bubbles are not provided (boolean algebra shows that an even number of input inversion bubbles is ignored, while an odd number of input inversion bubbles is transformed into an inversion bubble on the output).

The OTHER library has a 3-input majority gate, MAJ3i0, which proves useful in arithmetic circuits. Its output is asserted if two or three of its inputs are asserted.

The OTHER library includes buffers and inverters, BUFF and INV. Two buffers or inverters can be implemented in a single logic cell.

NOTE: Inverters are rarely required. The SOP16i7 macro uses the pASIC3, QuickRAM, QuickPCI, or Eclipse logic cell and therefore can be used only in pASIC3, QuickRAM, QuickPCI, or Eclipse designs.

The QuickLogic Macro Library has two useful sum-of-products macros, SOP14i7 and SOP16i7, which demonstrate the versatility of the logic cell. The SOP16i7 macro uses the pASIC3 or QuickRAM logic cell and therefore can be used only in pASIC3 or QuickRAM designs. Most sum-of-products circuits are handled automatically through the Logic Optimizer, as discussed in the Logic Optimizer chapter.

The library also includes a 2-to-4 decoder, DEC2t4, which requires only a single logic cell.

B.15.1 Macro Descriptions

B.15.1.1 XOR2i0

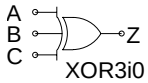
INPUTS: A, B

OUTPUTS: Z

Table B-86: XOR2i0

A	B	Z
L	L	L
L	H	H
H	L	H
H	H	L

B.15.1.2 XOR3i0



INPUTS: A, B, C

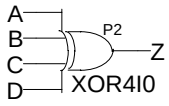
OUTPUTS: Z

Table B-87: XOR3i0

A	B	C	Z
L	L	L	L
L	L	H	H
L	H	L	H
L	H	H	L
H	L	L	H
H	L	H	L
H	H	L	L
H	H	H	H

B.15.1.3 XOR4i0

NOTE: This macro can only be used in pASIC3, QuickRAM, QuickPCI, and Eclipse designs.



INPUTS: A, B, C, D

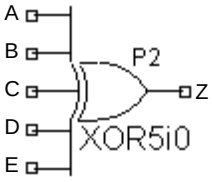
OUTPUTS: Z

Table B-88: XOR4i0

A, B, C, D	Z
Even number of inputs set to 1	L
Odd number of inputs set to 1	H

B.15.1.4 XOR5i0

NOTE: This macro can only be used in pASIC3, QuickRAM, QuickPCI, and Eclipse designs.



INPUTS: A, B, C, D, E

OUTPUTS: Z

© 2008 QuickLogic Corporation

Table B-89: XOR5i0

A, B, C, D, E	Z
Even number of inputs set to 1	L
Odd number of inputs set to 1	H

B.15.1.5 XNOR2i0

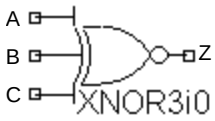
INPUTS: A, B

OUTPUTS: Z

Table B-90: XNOR2i0

A	B	Z
L	L	H
L	H	L
H	L	L
H	H	H

B.15.1.6 XNOR3i0



INPUTS: A, B, C

OUTPUTS: Z

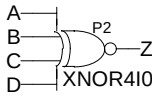
Table B-91: XNOR3i0

A	B	C	Z
L	L	L	H
L	L	H	L
L	H	L	L
L	H	H	H
H	L	L	L
H	L	H	H
H	H	L	H
H	H	H	L

© 2008 QuickLogic Corporation

B.15.1.7 XNOR4i0

NOTE: This macro can only be used in pASIC 3 and QuickRAM designs.



INPUTS: A, B, C, D

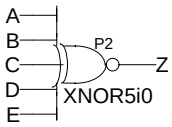
OUTPUTS: Z

Table B-92: XNOR4i0

A, B, C, D	Z
Even number of inputs set to 1	H
Odd number of inputs set to 1	L

B.15.1.8 XNOR5i0

NOTE: This macro can only be used in pASIC 3 and QuickRAM designs.



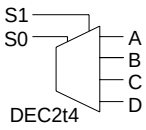
INPUTS: A, B, C, D, E

OUTPUTS: Z

Table B-93: XNOR5i0

A, B, C, D, E	Z
Even number of inputs set to 1	H
Odd number of inputs set to 1	L

B.15.1.9 DEC2t4



INPUTS: S1, S0

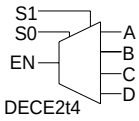
OUTPUTS: A, B, C, D

© 2008 QuickLogic Corporation

Table B-94: DEC2t4

S1	S0	A	B	C	D
L	L	H	L	L	L
L	H	L	H	L	L
H	L	L	L	H	L
H	H	L	L	L	H

B.15.1.10 DECE2t4



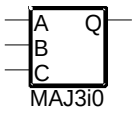
INPUTS: S1, S0, EN

OUTPUTS: A, B, C, D

Table B-95: DECE2t4

S1	S0	EN	A	B	C	D
X	X	L	L	L	L	L
L	L	H	H	L	L	L
L	H	H	L	H	L	L
H	L	H	L	L	H	L
H	H	H	L	L	L	H

B.15.1.11 MAJ3i0



INPUTS: A, B, C

OUTPUTS: Q

Table B-96: MAJ3i0

A	B	C	Q
L	L	L	L
L	L	H	L
L	H	L	L
L	H	H	H
H	L	L	L

Table B-96: MAJ3i0 (Continued)

A	B	C	Q
H	L	H	H
H	H	L	H
H	H	H	H

© 2008 QuickLogic Corporation

B.16 RAM Blocks

QuickLogic's QuickRAM device family includes embedded RAM blocks. Each block contains up to 1152 bits of RAM configurable in four modes:

- 64 deep by 18 bits wide (64 × 18)
- 128 deep by 9 bits wide (128 × 9)
- 256 deep by 4 bits wide (256 × 4)
- 512 deep by 2 bits wide (512 × 2)

Each of the four configurations uses only 1 RAM block. The number of RAM blocks varies by device, from 10 in the QL4016 to 22 in the QL4090.

Each RAM block has two ports: one for writes and one for reads. The RAM blocks have true dual-port functionality, allowing you to write and read from the RAM at the same time.

Each RAM block also has an `ASYNCRD` input. This input controls whether the read data port is updated on the positive edge of `RCLK` (synchronous reads) or when the read address bus, `RA`, changes (asynchronous reads). Setting the `ASYNCRD` pin high causes the read operation to be asynchronous. The `ASYNCRD` input must be tied to `VCC` or `GND` in the design.

NOTE: These macros can only be used in QuickRAM, QuickPCI and Eclipse family of devices.

B.16.1 Macro Descriptions

B.16.1.1 RAM64 x 18

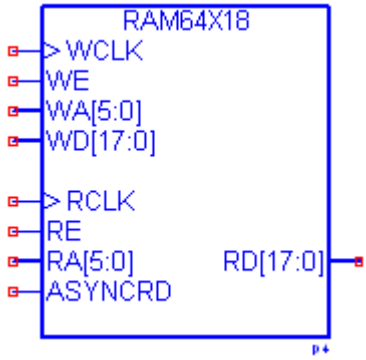


Table B-97: RAM Macros

Macro	Description	Depth	Width	Total Bits	RAM blocks
RAM64 X 18	Dual-Port RAM	64	18	1152	1
RAM128 X 9	Dual-Port RAM	128	9	1152	1
RAM256 X 4	Dual-Port RAM	256	4	1024	1
RAM512 X 2	Dual-Port RAM	512	2	1024	1

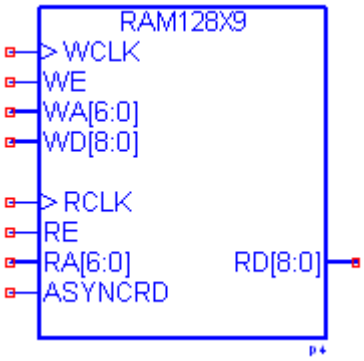
© 2008 QuickLogic Corporation

INPUTS: WCLK, WA[5:0], WD[17:0]. WE, RCLK, RA[5:0], RE, ASYNCRD
OUTPUTS: RD[17:0]

Table B-98: RAM64 X 18

WCLK	WE	WA[5:0]	WD[17:0]	RCLK	RE	RA[5:0]	ASYNCRD	RD[17:0]	internal ram
^	1	ADDR	DATA	X	X	X	X	X	ram(ADDR)= DATA
X	0	X	X	X	X	X	X	X	ram unchanged
X	X	X	X	X	1	ADDR	1	ram (ADDR)	
X	X	X	X	^	1	ADDR	0	ram (ADDR)	
X	X	X	X	X	0	X	X	hold	
^	1	ADDR	DATA	[Last clocked RA = WA]			X	DATA	ram(ADDR)= DATA

B.16.1.2 RAM128 x 9



INPUTS: WCLK, WA[6:0], WD[8:0]. WE, RCLK, RA[6:0], RE, ASYNCRD

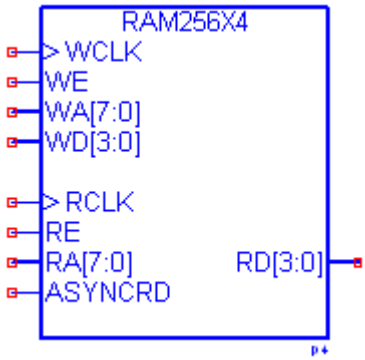
OUTPUTS: RD[8:0]

Table B-99: RAM128 X 9

WCLK	WE	WA[6:0]	WD[8:0]	RCLK	RE	RA[6:0]	ASYNCRD	RD[8:0]	internal ram
^	1	ADDR	DATA	X	X	X	X	X	ram(ADDR)=DATA
X	0	X	X	X	X	X	X	X	ram unchanged
X	X	X	X	X	1	ADDR	1	ram(ADDR)	
X	X	X	X	^	1	ADDR	0	ram(ADDR)	
X	X	X	X	X	0	X	X	hold	
^	1	ADDR	DATA	[Last clocked RA = WA]			X	DATA	ram(ADDR)=DATA

© 2008 QuickLogic Corporation

B.16.1.3 RAM256 x 4

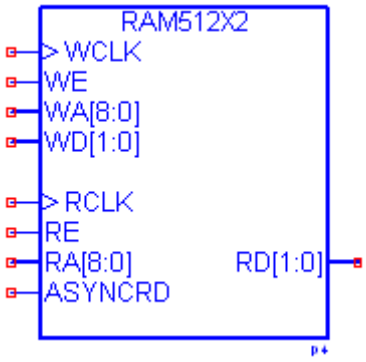


inputs: WCLK, WA[7:0], WD[3:0]. WE, RCLK, RA[7:0], RE, ASYNCRD
outputs: RD[3:0]

Table B-100: RAM256 X 4

WCLK	WE	WA[7:0]	WD[3:0]	RCLK	RE	RA[7:0]	ASYNCRD	RD[3:0]	internal ram
^	1	ADDR	DATA	X	X	X	X	X	ram(ADDR)=DATA
X	0	X	X	X	X	X	X	X	ram unchanged
X	X	X	X	X	1	ADDR	1	ram(ADDR)	
X	X	X	X	^	1	ADDR	0	ram(ADDR)	
X	X	X	X	X	0	X	X	hold	
^	1	ADDR	DATA	[Last clocked RA = WA]			X	DATA	ram(ADDR)=DATA

B.16.1.4 RAM512 x 2



© 2008 QuickLogic Corporation

INPUTS: WCLK, WA[8:0], WD[1:0]. WE, RCLK, RA[8:0], RE, ASYNCRD
OUTPUTS: RD[1:0]

Table B-101: RAM512 X 2

WCLK	WE	WA[8:0]	WD[1:0]	RCLK	RE	RA[8:0]	ASYNCRD	RD[1:0]	internal ram
^	1	ADDR	DATA	X	X	X	X	X	ram(ADDR)=DATA
X	0	X	X	X	X	X	X	X	ram unchanged
X	X	X	X	X	1	ADDR	1	ram(ADDR)	
X	X	X	X	^	1	ADDR	0	ram(ADDR)	
X	X	X	X	X	0	X	X	hold	
^	1	ADDR	DATA	[Last clocked RA = WA]			X	DATA	ram(ADDR)=DATA

© 2008 QuickLogic Corporation

B.17 RAM Blocks for 0.25 um Devices

QuickLogic's Eclipse device families includes embedded RAM blocks. Each block contains up to 2304 bits of RAM configurable in four modes:

- 128 deep by 18 bits wide (128 x 18)
- 256 deep by 9 bits wide (256 x 9)
- 512 deep by 4 bits wide (512 x 4)
- 1024 deep by 2 bits wide (1024 x 2)

Each of the four configurations uses only 1 RAM block. The number of RAM blocks varies by device, from 32 in QL6500 or QL7160 to 36 in QL6600 or QL7180.

Each RAM block has two ports: one for writes and one for reads. The RAM blocks have true dual-port functionality, allowing you to write and read from the RAM at the same time.

Each RAM block also has an ASYNCRD input. This input controls whether the read data port is updated on the positive edge of RCLK (synchronous reads) or when the read address bus, RA, changes (asynchronous reads). Setting the ASYNCRD pin high causes the read operation to be asynchronous. The ASYNCRD input must be tied to VCC or GND in the design.

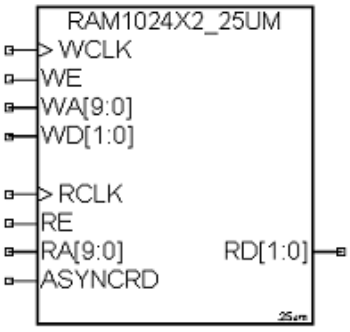
NOTE: These macros can only be used in Eclipse families of devices.

Table B-102: 0.25 um Devices RAM Macros

Macro	Description	Depth	Width	Total Bits	RAM Blocks
RAM128 X 18_25um	Dual-Port RAM for Eclipse Designs	128	18	2304	1
RAM256 X 9_25um	Dual-Port RAM for Eclipse Designs	256	9	2304	1
RAM512 X 4_25um	Dual-Port RAM for Eclipse Designs	512	4	2048	1
RAM1024 X 2_25um	Dual-Port RAM for Eclipse Designs	1024	2	2048	1

B.17.1 Macro Descriptions

B.17.1.1 RAM1024 x 2_25UM

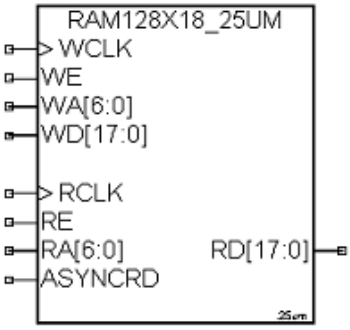


INPUTS: WCLK, WA[9:0], WD[1:0], WE, RCLK, RA[9:0], RE, ASYNCRD
OUTPUTS: RD[1:0]

Table B-103: RAM1024 X 2_25UM

WCLK	WE	WA[9:0]	WD[1:0]	RCLK	RE	RA[9:0]	ASYNCRD	RD[1:0]	internal RAM
^	1	ADDR	DATA	X	X	X	X	X	ram(ADDR)=DATA
X	0	X	X	X	X	X	X	X	ram unchanged
X	X	X	X	X	1	ADDR	1	ram(ADDR)	
X	X	X	X	^	1	ADDR	0	ram(ADDR)	
X	X	X	X	X	0	X	X	hold	
^	1	ADDR	DATA	[Last clocked RA = WA]			X	DATA	ram(ADDR)=DATA

B.17.1.2 RAM128 x 18_25UM



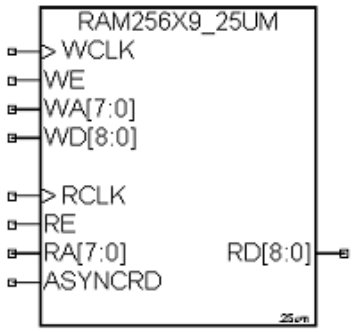
INPUTS: WCLK, WA[6:0], WD[17:0], WE, RCLK, RA[6:0], RE, ASYNCRD
OUTPUTS: RD[17:0]

Table B-104: RAM128 X 18_25UM

WCLK	WE	WA[6:0]	WD[17:0]	RCLK	RE	RA[6:0]	ASYNCRD	RD[17:0]	internal RAM
^	1	ADDR	DATA	X	X	X	X	X	ram(ADDR)=DATA
X	0	X	X	X	X	X	X	X	ram unchanged
X	X	X	X	X	1	ADDR	1	ram(ADDR)	
X	X	X	X	^	1	ADDR	0	ram(ADDR)	
X	X	X	X	X	0	X	X	hold	
^	1	ADDR	DATA	[Last clocked RA = WA]			X	DATA	ram(ADDR)=DATA

© 2008 QuickLogic Corporation

B.17.1.3 RAM256 x 9_25UM

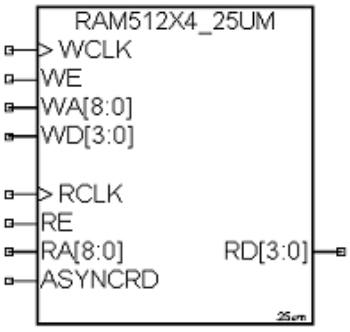


INPUTS: WCLK, WA[7:0], WD[8:0], WE, RCLK, RA[7:0], RE, ASYNCRD
OUTPUTS: RD[8:0]

Table B-105: RAM256 X 9_25UM

WCLK	WE	WA[7:0]	WD[8:0]	RCLK	RE	RA[7:0]	ASYNCRD	RD[8:0]	internal RAM
^	1	ADDR	DATA	X	X	X	X	X	ram(ADDR)=DATA
X	0	X	X	X	X	X	X	X	ram unchanged
X	X	X	X	X	1	ADDR	1	ram(ADDR)	
X	X	X	X	^	1	ADDR	0	ram(ADDR)	
X	X	X	X	X	0	X	X	hold	
^	1	ADDR	DATA	[Last clocked RA = WA]			X	DATA	ram(ADDR)=DATA

B.17.1.4 RAM512 x 4_25UM



© 2008 QuickLogic Corporation

INPUTS: WCLK, WA[8:0], WD[3:0], WE, RCLK, RA[8:0], RE, ASYNCRD
OUTPUTS: RD[3:0]

Table B-106: RAM512 X 4_25UM

WCLK	WE	WA[8:0]	WD[3:0]	RCLK	RE	RA[8:0]	ASYNCRD	RD[3:0]	internal RAM
^	1	ADDR	DATA	X	X	X	X	X	ram(ADDR)= DATA
X	0	X	X	X	X	X	X	X	ram unchanged
X	X	X	X	X	1	ADDR	1	ram(ADDR)	
X	X	X	X	^	1	ADDR	0	ram(ADDR)	
X	X	X	X	X	0	X	X	hold	
^	1	ADDR	DATA	[Last clocked RA = WA]	X	DATA	ram(ADDR)=DATA		

B.18 RAM Blocks for PolarPro Devices

The QuickLogic PolarPro device family includes embedded RAM blocks. Each block contains up to 4 K bits of memory.

Each RAM block has two ports: one for write and one for read. The RAM blocks have true dual-port functionality, allowing users to simultaneously write to and read from the RAM.

The RAM block can read/write in asynchronous mode only when used as a FIFO.

NOTE: These macros can only be used in the PolarPro family of devices.

Table B-107 lists the PolarPro device RAM macros.

Table B-107: PolarPro Devices RAM Macros

Macro	Description	Total Bits	RAM Blocks
RAM4K	RAM for PolarPro Designs	4,096	1
RAM8K_hc	Two 4 K rams in horizontal concatenation for PolarPro Designs	8,192	2
RAM8K_vc	Two 4 K rams in vertical concatenation for PolarPro Designs	8,192	2
RAM8K_vc_dp	Two 4 K rams in vertical concatenation acting as dual port for PolarPro Designs	8,192	2

NOTE: When the PolarPro RAM is used for different read and write data widths the extra bits must be connected to GND correctly (i.e., not the higher bits, but the bits as the read/write data widths). The RRW generated generic model takes care of this issue, but when the designer uses the macros the designer must take care of this issue.

© 2008 QuickLogic Corporation

B.19 RAM Blocks for PolarPro II Devices

The QuickLogic PolarPro II device family includes two types of embedded RAM blocks. Each block contains 2 K/4 K bits of memory.

Each RAM block has two ports: one for write and one for read. The RAM blocks have true dual-port functionality, allowing users to simultaneously write to and read from the RAM.

The RAM block can read/write in asynchronous mode only when used as a FIFO. There is a port (DIR) to control the FIFO operation as a PUSH or POP on each port.

NOTE: These macros can only be used in the PolarPro II family of devices.

Table B-108 lists the PolarPro II device RAM macros.

Table B-108: PolarPro II Device RAM Macros

Macro	Description	Total Bits	RAM Blocks
RAM2K	2 K RAM for PolarPro II designs	2,048	1
RAM4K_hc	Two 2 K RAMs in horizontal concatenation for PolarPro II designs	4,096	2
RAM4K_vc	Two 2 K RAMs in vertical concatenation for PolarPro II designs	4,096	2
RAM4K_vc_dp	Two 2 K RAMs in vertical concatenation acting as dual port for PolarPro II designs	4,096	2
RAM4K	4 K RAM for PolarPro II designs	4,096	1
RAM8K_hc	Four 4 K RAMs in horizontal concatenation for PolarPro II designs	8,192	2
RAM8K_vc	Two 4 K RAMs in vertical concatenation for PolarPro II designs	8,192	2
RAM8K_vc_dp	Two 4 K RAMs in vertical concatenation acting as dual port for PolarPro II designs	8,192	2

© 2008 QuickLogic Corporation

B.20 PCI Interface Cores Macros

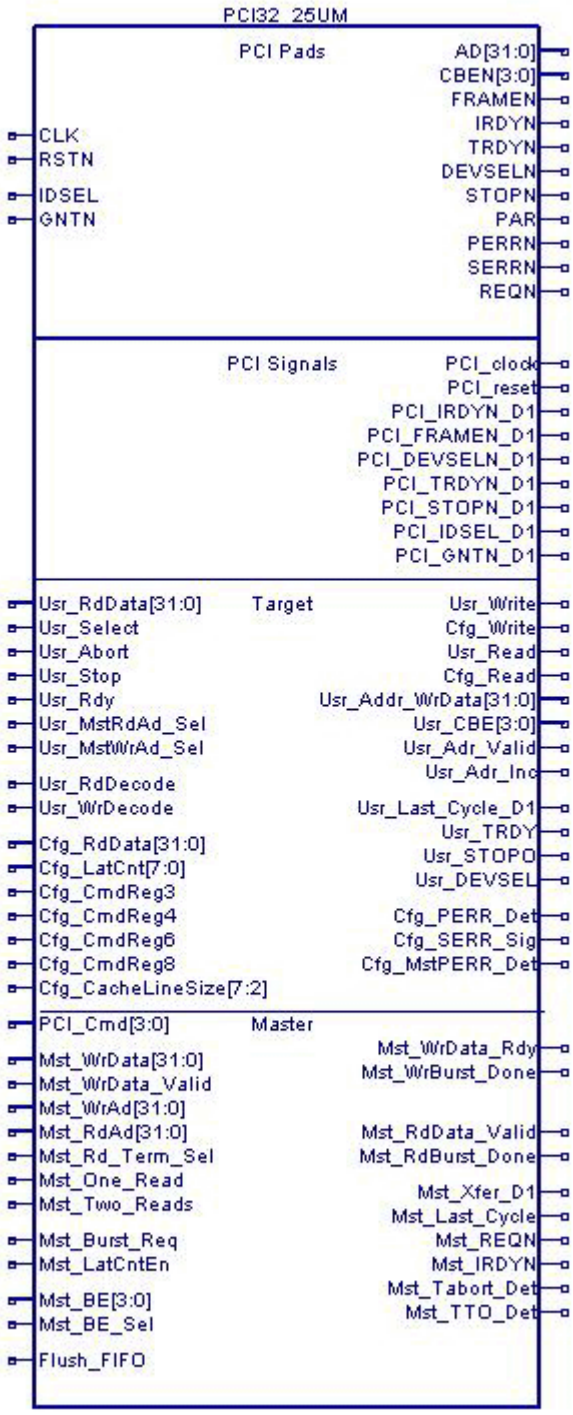
These PCI Interface macros are used to attach local interface programmable logic design to the PCI Cores. For more information, refer to QuickLogic QuickPCI documentation on the QuickLogic website. Some examples of PCI Interface Cores are given below.

© 2008 QuickLogic Corporation

B.20.1 Macro Descriptions

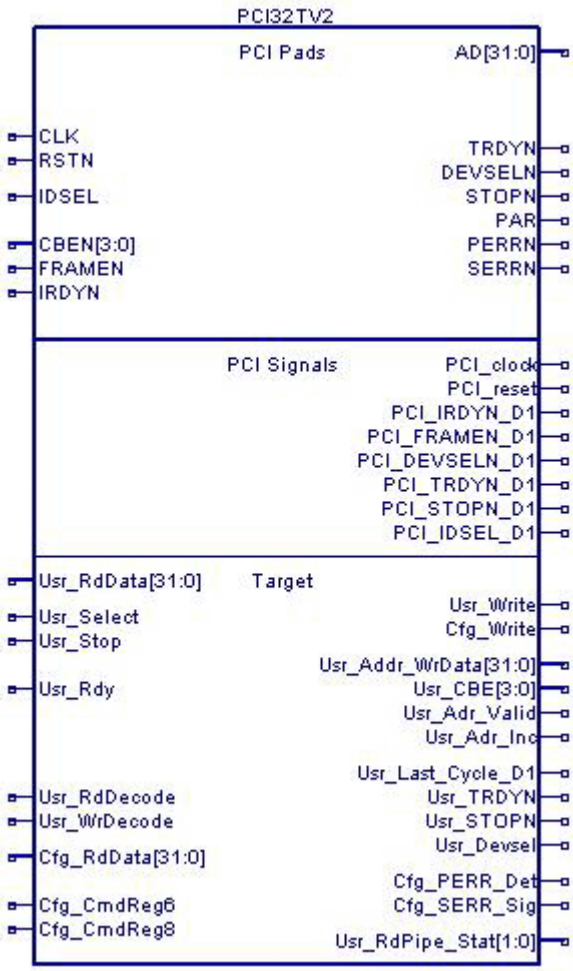
B.20.1.1 PCI32_25

NOTE: This macro can only be used in QuickPCI QL5632 and QL5732 designs.



B.20.1.2 PCI32TV2

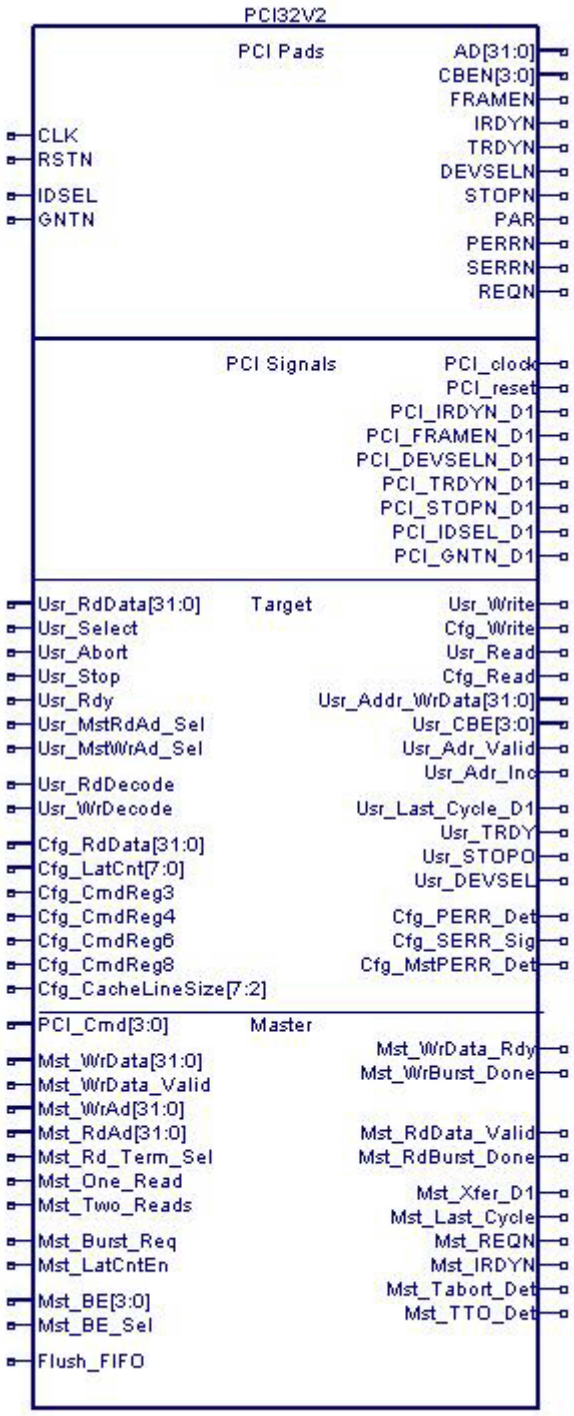
NOTE: This macro can only be used in QuickPCI QL5810, QL5820, and QL5840 designs.



© 2008 QuickLogic Corporation

B.20.1.3 PCI32V2

NOTE: This macro can only be used in QuickPCI QL5822 and QL5842 designs.



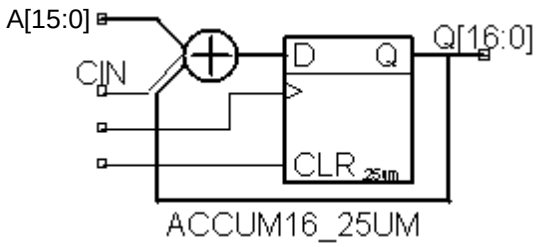
© 2008 QuickLogic Corporation

B.21 Embedded Computational Units (ECUs)

The ECU macros are designed to implement high-speed arithmetic operations, such as accumulation, addition (registered or unregistered), multiplication (registered or unregistered), multiply-accumulate, and multiply-add (registered or unregistered).

B.21.1 Macro Descriptions

B.21.1.1 ACCUM16_25UM

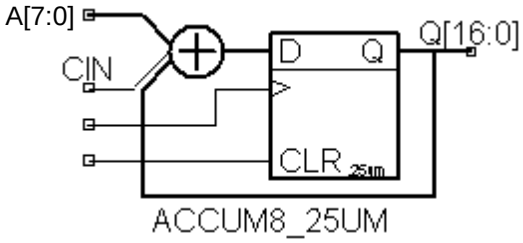


INPUTS: CLK, CLR, CIN, A[15:0]
OUTPUTS: Q[16:0]

Table B-109: ACCUM16_25UM

CLK	CLR	CIN	A[15:0]	Q[16:0]
X	H	X	X	0 0000 0000 0000 0000
L	L	X	X	Q[16:0]
H	L	X	X	Q[16:0]
First ^	L	L	A[15:0]	0, A[15:0]
First ^	L	H	A[15:0]	A[15:0] + 1
Second ^	L	L	A[15:0]	A[15:0] + Q[16:0]
Second ^	L	H	A[15:0]	A[15:0] + Q[16:0] + 1

B.21.1.2 ACCUM8_25UM



© 2008 QuickLogic Corporation

INPUTS: CLK, CLR, CIN, A[7:0]
OUTPUTS: Q[16:0]

Table B-110: ACCUM8_25UM

CLK	CLR	CIN	A[7:0]	Q[16:0]
X	H	X	X	0 0000 0000 0000 0000
L	L	X	X	Q[16:0]
H	L	X	X	Q[16:0]
First ^	L	L	A[7:0]	0 0000 0000, A[7:0]
First ^	L	H	A[7:0]	0 0000 000X, A[7:0] + 1
Second ^	L	L	A[7:0]	0, A[7:0] + Q[15:0]
Second ^	L	H	A[7:0]	0, A[7:0] + Q[15:0] + 1

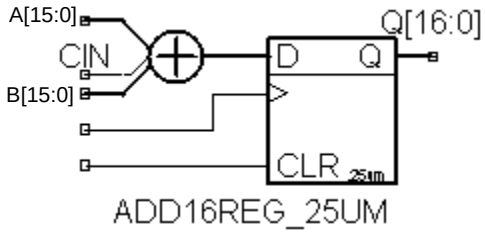
B.21.1.3 ADD16_25UM

INPUTS: CIN, A[15:0], B[15:0]
OUTPUTS: Q[16:0]

Table B-111: ADD16_25UM

CIN	A[15:0]	B[15:0]	Q[16:0]
L	A[15:0]	B[15:0]	A[15:0] + B[15:0]
H	A[15:0]	B[15:0]	A[15:0] + B[15:0] + 1

B.21.1.4 ADD16REG_25UM



INPUTS: CLK, CLR, CIN, A[15:0], B[15:0]
OUTPUTS: Q[16:0]

Table B-112: ADD16REG_25UM

CLK	CLR	CIN	A[15:0]	B[15:0]	Q[16:0]
X	H	X	X	X	0 0000 0000 0000 0000
L	L	X	X	X	Q[16:0]
H	L	X	X	X	Q[16:0]
^	L	L	A[15:0]	B[15:0]	A[15:0] + B[15:0]
^	L	H	A[15:0]	B[15:0]	A[15:0] + B[15:0] + 1

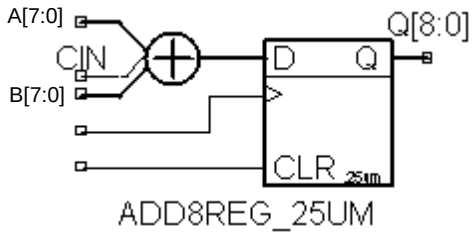
B.21.1.5 ADD16REG_25UM

INPUTS: CIN, A[7:0], B[7:0]
OUTPUTS: Q[8:0]

Table B-113: ADD16REG_25UM

CIN	A[7:0]	B[7:0]	Q[8:0]
L	A[7:0]	B[7:0]	A[7:0] + B[7:0]
H	A[7:0]	B[7:0]	A[7:0] + B[7:0] + 1

B.21.1.6 ADD8REG_25UM



INPUTS: CLK, CLR, CIN, A[7:0], B[7:0]
OUTPUTS: Q[8:0]

Table B-114: ADD8REG_25UM

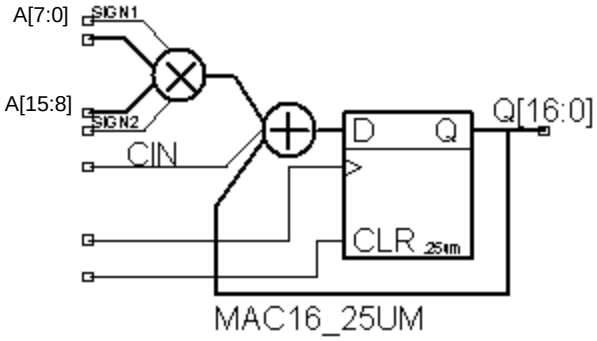
CLK	CLR	CIN	A[7:0]	B[7:0]	Q[8:0]
X	H	X	X	X	0 0000 0000
L	L	X	X	X	Q[8:0]

© 2008 QuickLogic Corporation

Table B-114: ADD8REG_25UM (Continued)

CLK	CLR	CIN	A[7:0]	B[7:0]	Q[8:0]
H	L	X	X	X	Q[8:0]
^	L	L	A[7:0]	B[7:0]	A[7:0] + B[7:0]
^	L	H	A[7:0]	B[7:0]	A[7:0] + B[7:0] + 1

B.21.1.7 MAC16_25UM



INPUTS: CLK, CLR, CIN, SIGN1, SIGN2, A[7:0], A[15:8]
OUTPUTS: Q[16:0]

Table B-115: MAC16_25UM

CLK	CLR	CIN	A[15:8]	A[7:0]	Q[16:0]
X	H	X	X	X	0 0000 0000 0000 0000
L	L	X	X	X	Q[16:0]
H	L	X	X	X	Q[16:0]
First ^	L	L	A[15:8]	A[7:0]	0, A[15:8] x A[7:0]
First ^	L	H	A[15:8]	A[7:0]	X, A[15:8] x A[7:0] + 1
Second ^	L	L	A[15:8]	A[7:0]	A[15:8] x A[7:0] + Q[15:0]

SIGN1 and SIGN2 must be tied to either GND ('0') or VCC ('1') based on the following table:

Table B-116: MAC16_25UM SIGN1 and SIGN2 Behaviour

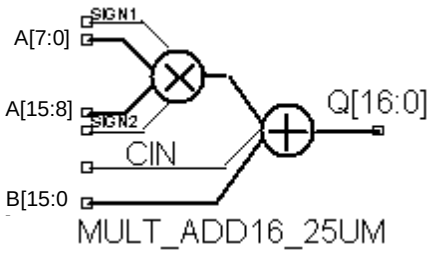
Sign-bit	Multiplier	Sign-bit	Multiplicand
SIGN1	A[7:0]	SIGN2	A[15:8]
'0	'(x00 - x7F) => (0 - 127)		
(x80 - xFF) => (128 - 255)	'0'	(x00 - x7F) => (0 - 127)	
(x80 - xFF) => (128 - 255)			
'1	'(x00 - x7F) => (0 - 127)		
(x80 - xFF) => (-128 - -1)	'1	'(x00 - x7F) => (0 - 127)	
(x80 - xFF) => (-128 - -1)			
'0	'(x00 - x7F) => (0 - 127)		

© 2008 QuickLogic Corporation

Table B-116: MAC16_25UM SIGN1 and SIGN2 Behaviour (Continued)

Sign-bit	Multiplier	Sign-bit	Multiplicand
(x80 - xFF) => (128 - 255)	'1	'(x00 - x7F) => (0 - 127)	
(x80 - xFF) => (-128 - -1)			
'1	'(x00 - x7F) => (0 - 127)		
(x80 - xFF) => (-128 - -1)	'0	'(x00 - x7F) => (0 - 127)	
(x80 - xFF) => (128 - 255)			

B.21.1.8 MULT_ADD16_25UM



INPUTS: CIN, SIGN1, SIGN2, A[7:0], A[15:8], B[15:0]
OUTPUTS: Q[16:0]

Table B-117: MULT_ADD16_25UM

CIN	A[15:8]	A[7:0]	B[15:0]	Q[16:0]
L	A[15:8]	A[7:0]	B[15:0]	A[15:8] x A[7:0] + B[15:0]
H	A[15:8]	A[7:0]	B[15:0]	A[15:8] x A[7:0] + B[15:0] + 1

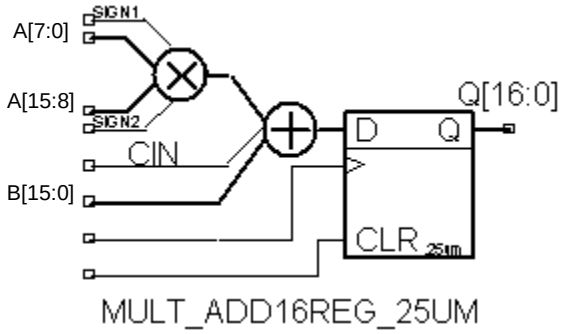
© 2008 QuickLogic Corporation

SIGN1 and SIGN2 must be tied to either GND (0) or VCC (1) based on **Table B-118**.

Table B-118: MULT_ADD16_25UM SIGN1 and SIGN2 Behaviour

Sign-bit	Multiplier	Sign-bit	Multiplicand
SIGN1	A[7:0]	SIGN2	A[15:8]
0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)	0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)
1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)	1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)
0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)	1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)
1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)	0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)

B.21.1.9 MULT_ADD16REG_25UM



INPUTS: CLK, CLR, CIN, SIGN1, SIGN2, A[7:0], A[15:8], B[15:0]
OUTPUTS: Q[16:0]

Table B-119: MULT_ADD16REG_25UM

CLK	CLR	CIN	A[15:8]	A[7:0]	B[15:0]	Q[16:0]
X	H	X	X	X	X	0 0000 0000 0000 0000
L	L	X	X	X	X	Q[16:0]
H	L	X	X	X	X	Q[16:0]
^	L	L	A[15:8]	A[7:0]	B[15:0]	A[15:8] x A[7:0] + B[15:0]
^	L	L	A[15:8]	A[7:0]	B[15:0]	A[15:8] x A[7:0] + B[15:0]
^	L	L	A[15:8]	A[7:0]	B[15:0]	A[15:8] x A[7:0] + B[15:0]
^	L	L	A[15:8]	A[7:0]	B[15:0]	A[15:8] x A[7:0] + B[15:0]
^	L	H	A[15:8]	A[7:0]	B[15:0]	A[15:8] x A[7:0] + B[15:0] + 1

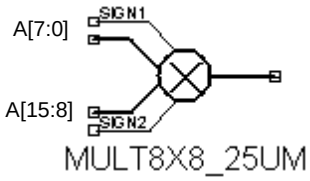
© 2008 QuickLogic Corporation

SIGN1 and SIGN2 must be tied to either GND (0) or VCC (1) based on **Table B-120**.

Table B-120: MULT_ADD16REG_25UM SIGN1 and SIGN2 Behaviour

Sign-bit	Multiplier	Sign-bit	Multiplicand
SIGN1	A[7:0]	SIGN2	A[15:8]
0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)	0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)
1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)	1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)
0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)	1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)
1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)	0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)

B.21.1.10 MULT8x8_25UM



INPUTS: SIGN1, SIGN2, A[7:0], A[15:8]
OUTPUTS: Q[15:0]

Table B-121: MULT8X8_25UM

A[7:0]	A[15:8]	Q[15:0]
A[7:0]	A[15:8]	A[15:8] x A[7:0]

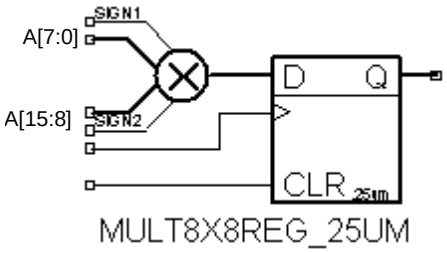
SIGN1 and SIGN2 must be tied to either GND ('0') or VCC ('1') based on **Table B-122 on page 480**.

Table B-122: MULT8X8_25UM SIGN1 and SIGN2 Behaviour

Sign-bit	Multiplier	Sign-bit	Multiplicand
SIGN1	A[7:0]	SIGN2	A[15:8]
'0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)	'0'	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)
'1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)	'1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)
'0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)	'1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)
'1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)	'0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)

© 2008 QuickLogic Corporation

B.21.1.11 MULT8X8REG_25UM



INPUTS: CLK, CLR, SIGN1, SIGN2, A[7:0], A[15:8]
OUTPUTS: Q[15:0]

Table B-123: MULT8X8REG_25UM

CLK	CLR	A[15:8]	A[7:0]	Q[15:0]
X	H	X	X	0000 0000 0000 0000
L	L	X	X	Q[15:0]
H	L	X	X	Q[15:0]
^	L	A[15:8]	A[7:0]	A[15:8] x A[7:0]

SIGN1 and SIGN2 must be tied to either GND (0) or VCC (1) based on **Table B-124 on page 481**.

Table B-124: MULT8X8REG_25UM SIGN1 and SIGN2 Behaviour

Sign-bit	Multiplier	Sign-bit	Multiplicand
SIGN1	A[7:0]	SIGN2	A[15:8]
0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)	0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)
1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)	1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)
0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)	1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)
1	(x00 - x7F) => (0 - 127) (x80 - xFF) => (-128 - -1)	0	(x00 - x7F) => (0 - 127) (x80 - xFF) => (128 - 255)

© 2008 QuickLogic Corporation

B.22 Low Voltage Differential Signal (LVDS)

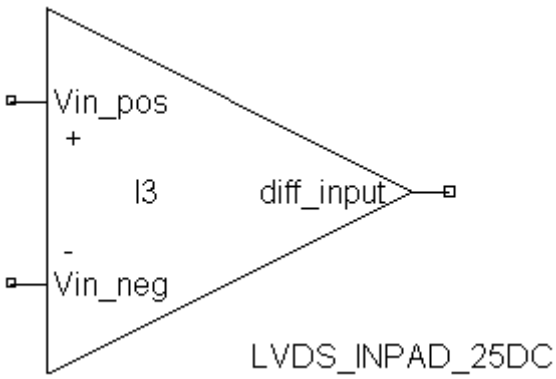
NOTE: LVDS macros can only be used in Eclipse devices.

The Eclipse Input/Outputs are separated into eight banks with each individual bank configurable to 2.5 V or 3.3 V operation using the VCCIO pins.

For information on how to interface Eclipse to Low Voltage Differential Signal (LVDS) Receiver, refer to *Application Note #36* on the QuickLogic website (www.quicklogic.com).

B.22.1 Macro Descriptions

B.22.1.1 LVDS_INPAD_25DC



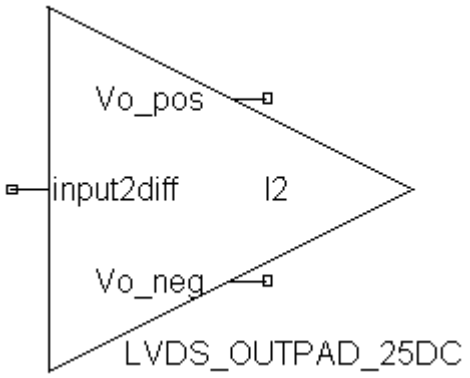
INPUTS: VIN_POS, VIN_NEG
OUTPUT: DIFF_INPUT

Table B-125: LVDS_INPAD_25DC

Vin_pos	Vin_neg	diff_input
L	L	L
L	H	L
H	L	H
H	H	L

© 2008 QuickLogic Corporation

B.22.1.2 LVDS_OUTPAD_25DC

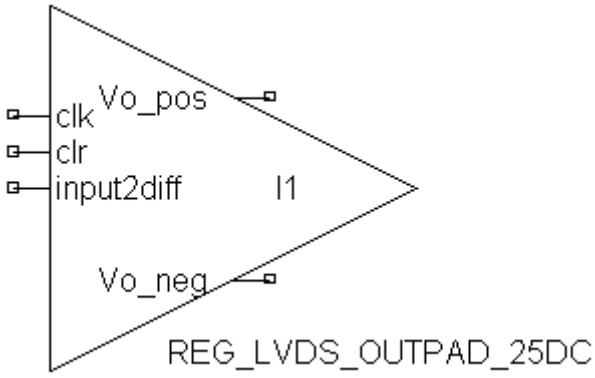


INPUT: INPUT2DIFF
OUTPUTS: VO_POS, VO_NEG

Table B-126: LVDS_OUTPAD_25DC

input2diff	Vo_pos	Vo_neg
L	L	H
H	H	L

B.22.1.3 REG_LVDS_OUTPAD_25DC



INPUTS: CLK, CLR, INPUT2DIFF
OUTPUTS: VO_POS, VO_NEG

Table B-127: REG_LVDS_OUTPAD_25DC

clk	clr	input2diff	Vo_pos	Vo_neg
X	H	X	L	L
L	L	X	Vo_pos	Vo_neg

© 2008 QuickLogic Corporation

Table B-127: REG_LVDS_OUTPAD_25DC

clk	clr	input2diff	Vo_pos	Vo_neg
H	L	X	Vo_pos	Vo_neg
^	L	L	L	H
^	L	H	H	L

© 2008 QuickLogic Corporation

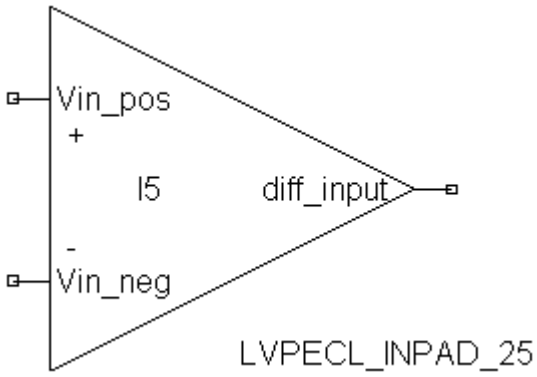
B.23 Low Voltage Positive Emitter Coupled Logic (LVPECL)

NOTE: LVPECL macros can only be used in Eclipse devices.

For information on how Eclipse Devices Support High-Speed LVPECL Transmission, refer to *Application Note #35* on the QuickLogic website (www.quicklogic.com).

B.23.1 Macro Descriptions

B.23.1.1 LVPECL_INPAD_25

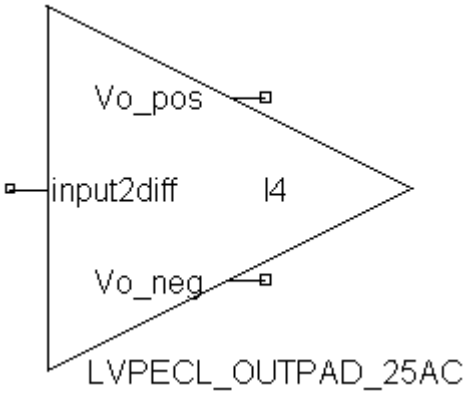


INPUTS: VIN_POS, VIN_NEG
OUTPUTS: DIFF_INPUT

Table B-128: LVPECL_INPAD_25

Vin_pos	Vin_neg	diff_input
L	L	L
L	H	L
H	L	H
H	H	L

B.23.1.2 LVPECL_OUTPAD_25AC

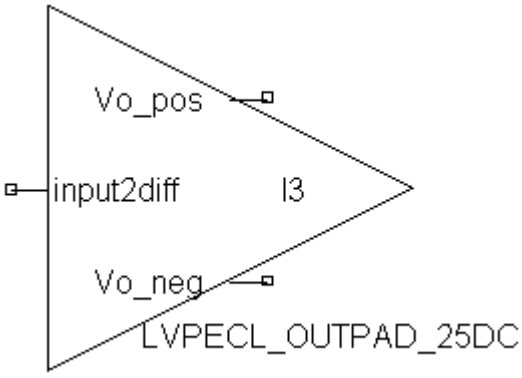


INPUTS: INPUT2DIFF
OUTPUTS: VO_POS, VO_NEG

Table B-129: LVPECL_OUTPAD_25AC

input2diff	Vo_pos	Vo_neg
L	L	H
H	H	L

B.23.1.3 LVPECL_OUTPAD_25DC



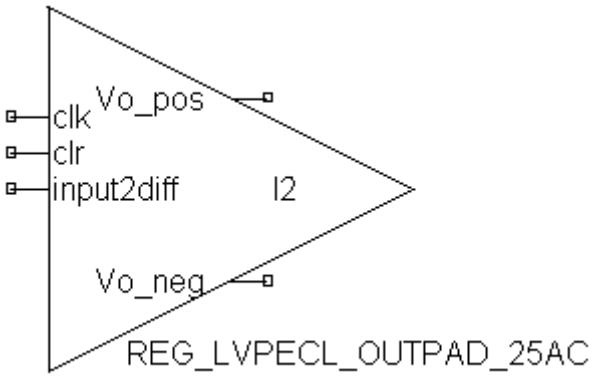
INPUTS: INPUT2DIFF
OUTPUTS: VO_POS, VO_NEG

Table B-130: LVPECL_OUTPAD_25DC

input2diff	Vo_pos	Vo_neg
L	L	H
H	H	L

© 2008 QuickLogic Corporation

B.23.1.4 REG_LVPECL_OUTPAD_25AC

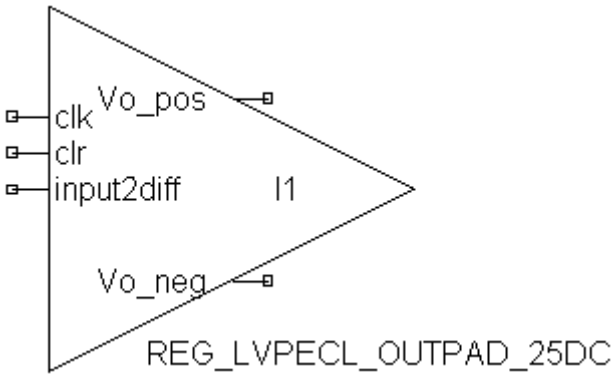


INPUTS: CLK, CLR, INPUT2DIFF
OUTPUTS: VO_POS, VO_NEG

Table B-131: REG_LVPECL_OUTPAD_25AC

clk	clr	input2diff	Vo_pos	Vo_neg
X	H	X	L	L
L	L	X	Vo_pos	Vo_neg
H	L	X	Vo_pos	Vo_neg
^	L	L	L	H
^	L	H	H	L

B.23.1.5 REG_LVPECL_OUTPAD_25DC



INPUTS: CLK, CLR, INPUT2DIFF
OUTPUTS: VO_POS, VO_NEG

Table B-132: REG_LVPECL_OUTPAD_25DC

clk	clr	input2diff	Vo_pos	Vo_neg
X	H	X	L	L
L	L	X	Vo_pos	Vo_neg
H	L	X	Vo_pos	Vo_neg
^	L	L	L	H
^	L	H	H	L

© 2008 QuickLogic Corporation

B.24 7400-Series TTL

The TTL library section includes more than 100 powerful TTL macros. The TTL library elements are listed in **Table B-133 on page 489**.

Table B-133: TLL Macros

Macro	Description
TTL02	quad 2-input NOR gates
TTL04	hex inverters
TTL08	quad 2-input AND gates
TTL11	triple 3-input AND gates
TTL21	dual 4-input AND gates
TTL27	triple 3-input NOR gates
TTL42q	4-to-10 decoder
TTL74q	dual D FFs with preset & clear
TTL77	4-bit D latch
TTL78q	dual J-K FFs with common clock & clear
TTL85	4-bit magnitude comparator
TTL86	quad 2-input XOR gates
TTL87	4-bit true/complement elements
TTL91	8-bit shift register
TTL98	4-bit data selector/storage register
TTL104q	gated J-K FF with preset & clear
TTL105q	gated J-K FF with preset & clear
TTL107q	dual J-K FFs with clear
TTL109q	dual J-K FFs with preset & clear
TTL116	dual 4-bit D latches with clear
TTL138q	3-to-8 decoder
TTL139q	dual 2-to-4 decoders
TTL145q	BCD to decimal decoder
TTL148	8-to-3 priority encoder
TTL150	16-to-1 multiplexer
TTL152	8-to-1 multiplexer
TTL153	dual 4-to-1 multiplexers
TTL154q	4-to-16 decoder
TTL157	quad 2-to-1 multiplexers
TTL161	4-bit binary counter with asynchronous clear
TTL163	4-bit binary counter with synchronous clear
TTL164q	8-bit parallel-out shift register

© 2008 QuickLogic Corporation

Table B-133: TLL Macros (Continued)

Macro	Description
TTL166q	8-bit parallel-load shift register
TTL169	4-bit binary up/down counter
TTL171q	quad D FFs with clear
TTL174q	hex D FFs with clear
TTL180	9-bit odd-even parity generator
TTL194q	4-bit bi-directional shift register
TTL240q	octal inverting tri-state drivers
TTL244q	octal non-inverting tri-state drivers
TTL259	8-bit addressable latches
TTL268q	hex D latches
TTL273q	octal D FFs with clear
TTL278	4-bit cascadable priority register
TTL279	quad S-R latches
TTL295q	4-bit right-shift left-shift register
TTL365	hex tri-state drivers
TTL366	hex inverting tri-state drivers
TTL367	hex tri-state drivers, 4-bit/2- bit banks
TTL368	hex inverting tri-state drivers, 4-bit/2-bit banks
TTL373q	octal D latches
TTL374q	octal D FFs
TTL375	4-bit latches with dual polarity outputs
TTL376q	quad J-K FFs with clear
TTL395q	4-bit cascadable shift register with clear
TTL396	octal storage register
TTL465	octal tri-state buffers
TTL466	octal inverting tri-state buffers
TTL467	octal tri-state buffers, 4-bit/4-bit banks
TTL468	octal inverting tri-state buffers, 4-bit/4-bit banks
TTL518	8-bit identity comparator
TTL594q	8-bit shift register with output register with clear
TTL595q	8-bit shift register with output register
TTL604q	octal 2-input multiplexed latches
TTL684	8-bit magnitude/identity comparitor
TTL686	8-bit magnitude/identity comparitor with enable
TTL688	8-bit identity comparitor
TTL821	10-bit FFs

© 2008 QuickLogic Corporation

Table B-133: TLL Macros (Continued)

Macro	Description
TTL822	10-bit inverting FFs
TTL823q	10-bit FFs with enable
TTL841q	10-bit latches
TTL842q	10-bit inverting latches

The *q* suffix indicates that the part is not an exact duplicate of the TTL device. The differences between these macros and their TTL counterparts are noted in **Table B-134 on page 491**.

Table B-134: Differences between TLLq Macros and TLL Counterparts

Macro	Difference between macro and actual TTL part
TTL42q	The TTL42q is a 4-line to 10-line BCD to decimal decoder and is similar to the 7442. The TTL42q has active high outputs, where the 7442 has active low outputs.
TTL74q	The TTL74q is a pair of d-type positive edge triggered flip-flops and is similar to the 7474. The TTL74q has active high clear and preset inputs, where these signals are active low on the 7474. Additionally, the TTL74q has only a true output for each F/F where the 7474 has both the true and complement outputs available.
TTL78q	The TTL78q is a pair of J-K flip-flops with individual presets, common clock and common clear, that are similar to the 7478. The TTL78 has active high clear and preset inputs, where these inputs are active low on the 7478. Additionally, the TTL78q has only a true output for each F/F where the 7478 has both the true and complement outputs available.
TTL104q	The TTL104q is a gated J-K Flip-Flop that is positive edge triggered. The TTL104q is similar to the 74104. The TTL104q has active high preset and clear inputs, where these inputs are active low on the 74104. Similarly, the J-K input on the TTL104q is active low, where it is active high on the 74104. Additionally, the TTL104q has a true output only while the 74104 has both true and complement.
TTL105q	The TTL105q is a gated J-K Flip-Flop that is positive edge triggered. The TTL105q is similar to the 74105. The TTL105q has active high preset and clear inputs, where these inputs are active low on the 74105. Additionally, the TTL105q has a true output only while the 74105 has both true and complement.
TTL107q	The TTL107q is a pair of positive edge triggered J-K flip-flops with independent clears and is similar to the 74107. The TTL107q has active high clear inputs that are active low on the 74107. Additionally, the TTL107q only has true outputs, while the 74107 has both true and complement.
TTL109q	The TTL109q is a pair of positive edge triggered J-K flip-flops with independent clear and preset functions. The TTL109q is similar to the 74109. The TTL109q has active high clear and preset inputs, that are active low on the 7410p. Additionally, the TTL109q only has true outputs, while the 74107 has both true and complement.
TTL138q	The TTL138q is a 3 to 8 bit decoder/de-multiplexer with enable, that is similar to the 74138. The TTL138q has a single active low enable, with active high outputs. The 74138 has a gated enable function, with active low outputs.
TTL139q	The TTL139q consists of two independent 2-line to 4-line de-multiplexers with enable. The TTL139q is similar to the 74139, having active high outputs, where the 74139 has active low.
TTL145q	The TTL145q is a BCD to decimal decoder that is similar to the 74145. The TTL139q has active high outputs, where the 74139 has active low outputs.
TTL154q	The TTL154q is a 4-line to 16-line Decoder that is similar to the 74154. The TTL154q has active high outputs, where the 74154 outputs are active low.

© 2008 QuickLogic Corporation

Table B-134: Differences between TLLq Macros and TLL Counterparts (Continued)

Macro	Difference between macro and actual TTL part
TTL164q	The TTL164q is an 8-bit parallel output serial shift register that is similar to the 74164. The TTL164q has an active high clear that is active low on the 74164.
TTL166q	The TTL166q is an 8-bit parallel load shift register that is similar to the 74166. The TTL166q has an active high clear that is active low on the 74166. Additionally, the TTL166q does not have a clock inhibit function.
TTL171q	The TTL171q is a 4-bit register with common clear and clock that is similar to the 74171. The TTL171q has an active high clear, and only has true outputs.
TTL174q	The TTL174q is a 6-bit register with common clear and clock that is similar to the 74174. The TTL174q has an active high clear where the 74174 has an active low clear.
TTL194q	The TTL194q is a 4-bit bi-directional shift register with clear that is similar to the 74194. The TTL194q has an active high clear where the 74194 has an active low clear.
TTL240q	The TTL240q is an inverting, octal tri-state buffer that is enabled in two groups of four bits. The TTL240q is similar to the 74240, having active high enables where the 74240 has active low enables. Note that the macro's tri-state outputs are available only as I/O pads, not internal to the chip.
TTL244q	The TTL244q is an octal tri-state buffer that is enabled in two groups of four bits. The TTL244q is similar to the 74244, having active high enables where the 74244 has active low enables. Note that the macro's tri-state outputs are available only as I/O pads, not internal to the chip.
TTL268q	The TTL268q is a hex D-type transparent latch that is similar to the 74268 that has tri-state outputs.
TTL273q	The TTL273q is an octal D-type flip-flop with clear that is similar to the 74273. The TTL273q has an active high clear where the 74273 has an active low clear.
TTL295q	The 74295q is a 4-bit shift-right/left register that is similar to the 74295, that has tri-state outputs.
TTL373q	The TTL373q is an octal D-type transparent latch that is similar to the 74373 that has tri-state outputs.
TTL374q	The TTL374q is an octal D-type register that is similar to the 74374 that has tri-state outputs.
TTL376q	The TTL376q is a 4-bit J-K' set of flip flops with common clear and clock. The TTL376q is similar to the 74376, having an active high clear input instead of active low.
TTL395q	The TTL395q is a 4-bit loadable and cascadable shift register with clear. The TTL395q is similar to the 74395, having an active high clear input instead of active low, and it does not have tri-state outputs.
TTL594q	The TTL594q is an 8-bit shift register with output register and is similar to the 74594. The TTL594q differs in that the clear functions are both active high.
TTL595q	The TTL595q is an 8-bit shift register with output register and is similar to the 74595. The TTL595q differs in that the clear functions are both active high, and it does not have tri-state outputs.
TTL604q	The TTL604q is a pair of octal storage registers that have their outputs muxed together. This function is similar to the 74604, except the TTL604q does not have tri-state outputs.
TTL823q	The TTL823q is an 10-bit storage register with clock enable that is similar to the 74823 that has tri-state outputs.
TTL841q	The TTL841q is a 10-bit flow-through latch that is similar to the 74841 that has tri-state outputs.
TTL842q	The TTL842q is a 10-bit flow-through latch with inverting data inputs, that is similar to the 74841 that has tri-state outputs.

© 2008 QuickLogic Corporation

Some of the non-q TTL macros have slight differences from the TTL device. The differences between these macros and their TTL counterparts are noted in **Table B-135 on page 493**.

Table B-135: Differences between Non-q TLL Macros and TLL

Macro	Difference between macro and actual TTL part
TTL116	The macro has active high clear inputs; the TTL device has active low clear inputs.
TTL153	The macro has select inputs named S0 and S1; the TTL device has select inputs named A and B.
TTL161	The macro has active low enable inputs and ripple-carry output; the TTL device has active high enable inputs and ripple-carry output. The macro has active high clear inputs; the TTL device has active low clear inputs.
TTL163	The macro has active low enable inputs and ripple-carry output; the TTL device has active high enable inputs and ripple-carry output. The macro has active high clear inputs; the TTL device has active low clear inputs.

© 2008 QuickLogic Corporation

B.25 Master Cells

The pASIC logic cell, with its unprecedented flexibility can construct a selection of macros beyond the scope of the supplied library. For this reason, QuickLogic allows the designer to create new macros using the underlying pASIC building blocks—the master cells.

Master cells represent the underlying building blocks of a pASIC device. The master cell represents the lowest level in the design hierarchy.

NOTE: Each hard macro is built from at most one master cell. Logic macros use the logic master cell, while pad macros use either the bi-directional, input, or clock master cell. Note that hard-macro symbols must be specified as type CELL in the SCS Symbol Editor. This is done in the Symbol Editor, by selecting **Edit>Symbol Type**. Refer to QuickLogic's supplemental documentation for information about using hard macros with other schematic tools.

The pASIC architecture consists of one variety of logic cell and three varieties of I/O cells. The architecture is presented in detail in the *QuickLogic Databook*.

© 2008 QuickLogic Corporation

B.26 pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro and PolarPro II-Specific Macros

The *QuickLogic Macro Library* contains a set of pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro and PolarPro II-specific macros besides the generic ones mentioned earlier in this chapter.

The pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro and PolarPro II macros allow the explicit implementation of pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro and PolarPro II architectural features such as: the input/feedback registers, array clock networks, and global clock networks in schematic-based designs. If specific pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro and PolarPro II features are not required, the generic macros should be used. Designs using generic macros can be targeted for a pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro or PolarPro II device.

NOTE: For Verilog or VHDL designs, please refer to the chapter **"Placer" on page 143** for discussions on pad type implementations.

The following table contains all of the macros specific to pASIC3, QuickRAM, QuickPCI, and Eclipse, and the sections where they are described.

Table B-136: pASIC3, QuickRAM, QuickPCI, and Eclipse-Specific Macros

Macro Name	Library Reference Section
ADD16P2	Arithmetic
ADD32P2	Arithmetic
AND16i7	Gates
AND7i0	Gates
AND7i1	Gates
AND8i0	Gates
AND8i1	Gates
AND9i4	Gates
ECOMP16	Arithmetic
ECOMP2	Arithmetic
ECOMP32	Arithmetic
ECOMP4	Arithmetic
ECOMP8	Arithmetic
FSTADD16	Arithmetic
FSTADD4	Arithmetic
FSTADD8	Arithmetic
LOGIC2	Logic Cell Macro
LSHFT2Q2	Shift Registers
NAND15i6	Gates
NAND7i0	Gates
NAND8i0	Gates

© 2008 QuickLogic Corporation

Table B-136: pASIC3, QuickRAM, QuickPCI, and Eclipse-Specific Macros (Continued)

Macro Name	Library Reference Section
NOR16i9	Gates
NOR7i0	Gates
NOR9i5	Gates
OR15i8	Gates
SOP16i7	Combinatorial
SUB16P2	Arithmetic
SUB32P2	Arithmetic
UCT8P2	Counters
UCTE16P2	Counters
UCTX16P2	Counters
UDLF6	Counters
UDLF12	Counters
UDLF18	Counters
UDLF24	Counters
XNOR4i0	Combinatorial
XNOR5i0	Combinatorial
XOR4i0	Combinatorial
XOR5i0	Combinatorial

© 2008 QuickLogic Corporation

B.27 pASIC3, QuickRAM, and QuickPCI-Specific Macros

The *QuickLogic Macro Library* contains a set of pASIC3, QuickRAM, and QuickPCI-specific macros besides the generic ones mentioned earlier in this chapter.

The pASIC3, QuickRAM, and QuickPCI macros allow the explicit implementation of pASIC3, QuickRAM, and QuickPCI architectural features such as: the input/feedback registers, array clock networks, and global clock networks in schematic-based designs. If specific pASIC3, QuickRAM, and QuickPCI features are not required, the generic macros should be used. Designs using generic macros can be targeted for either a pASIC3, QuickRAM, or QuickPCI device. However, the pASIC3, QuickRAM, and QuickPCI-specific macros are not supported for Eclipse designs. The following table contains all of the macros specific to pASIC3, QuickRAM, and QuickPCI, and the sections where they are described.

Table B-137: pASIC3, QuickRAM, and QuickPCI-Specific Macros

Macro Name	Library Reference Section
BIiPADFF	I/O Macros
BIorPADF	I/O Macros
BIPADFF	I/O Macros
BPAD16F	I/O Macros
BPAD4FF	I/O Macros
CKdPADFF	I/O Macros
CKPADFF	I/O Macros
CKtPADFF	I/O Macros
GCLKBUFF	I/O Macros
HDdPADFF	I/O Macros
HDiPADFF	I/O Macros
HDPADFF	I/O Macros
INPADFF	I/O Macros
IPAD16FF	I/O Macros
IPAD4FF	I/O Macros
IPAD8FF	I/O Macros

B.27.1 Hard Macros

It is important to understand the difference between the two main groups of macros (*hard macros* and *master cells*) referenced in this section. The master cells are the lowest level primitives in the design hierarchy. They are usually found in the `MCELL` symbol library. The master cells are: LCELL2, BICELL2, INCELL2, and CKCELL2.

For more information, refer to **“pASIC3, QuickRAM, and QuickPCI-Specific Macros” on page 497**.

The master cells should only be used to construct hard macros, which fit within a logic cell or an I/O cell. Using the master cells, designers can add customized hard macros to those in the library.

© 2008 QuickLogic Corporation

Some of the types of hard macros include: I/O pads, AND/OR gates, and D/T/JK flip-flops. These single-cell macros are used to build the entire library of *soft macros* including counters, adders, and comparators. When building your own hard macros out of master cells, be sure to change the symbol type to CELL using the Symbol Editor (use **Edit>Symbol Type**).

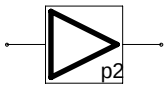
All generic hard/ soft macros and master cells in the QuickLogic Macro Library can be used for pASIC3, QuickRAM, and QuickPCI designs. However, specially designed hard macros and master cells are available to facilitate the direct implementation of pASIC3, QuickRAM, and QuickPCI-specific features. The initial sections provide information on the three types of hard macros: `Logic Cell`, `GCLK Buffer`, and `I/O Pads`. The latter sections contain information on the `Master Logic Cell` and `Master I/O Cells`.

B.27.1.1 Logic Cell Macro (LOGIC2)

The pASIC3, QuickRAM, and QuickPCI devices support two different clock networks: array and global. This macro is found in the OTHER symbol library. Since `LOGIC2` is a hard macro (and not a master cell), it can be used more than once (i.e., multiple times) within a particular schematic hierarchical level.

B.27.1.2 Internal Global Network Buffer (GCLKBUFF)

Figure B-24: GCLKBUFF



The pASIC 3, and QuickRAM devices support two different clock networks: array and global. The array clock network can only be accessed from the external `ACLK/I` pins. On the other hand, the global clock network can either be driven by the external GCLK/I pins or internal nets. The `GCLKBUFF` macro is used to specify an internally generated global clock net. This macro is found in the OTHER symbol library. The number of GCLKBUFF macros allowed in each design depends upon the particular pASIC3, QuickRAM, or QuickPCI device and the number of global clock nets already used in GCLK/ I pins. For more information on the clock networks, please refer to **“Router” on page 155**.

B.27.1.3 I/O Macros

The pASIC3, QuickRAM, and QuickPCI I/O cells contain input/feedback registers. The registers are designed to be used as input registers (the non-registered input signal is still available) and/or as feedback registers when the `I/O` output enable signal is asserted. The input/feedback register is an edge-triggered D-type flip-flop with clock-enable and asynchronous clear controls. The clock, enable, and clear signals can be driven by the global clock networks or from any internal nets (this excludes the array clock network driven by the `ACLK/I` pins).

The Logic Optimizer in SpDE can automatically allocate resources to utilize these input/feedback registers. Alternatively, the designer may choose to explicitly use a specific I/O pad macro.

© 2008 QuickLogic Corporation

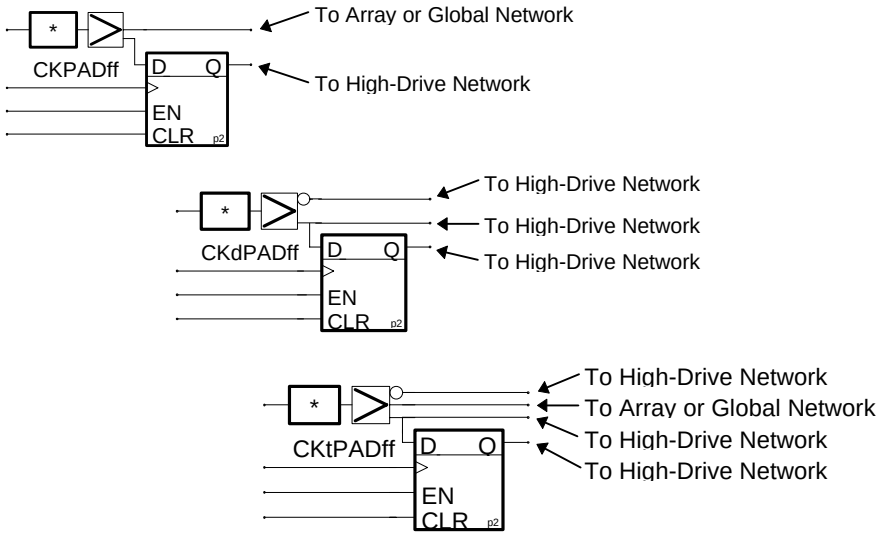
Table B-138 lists the generic pad macros with their pASIC3, QuickRAM, and QuickPCI equivalents. Multiple instance macros (e.g. `IPAD4ff` and `BPAD8ff`) are also supported, but they are not listed in the table.

Table B-138: Generic Pad Macros and the pASIC3, QuickRAM, and QuickPCI Equivalents

Generic PAD Macros	pASIC3, QuickRAM, and QuickPCI-Specific PADs
CKPAD	CKPADff
CKdPAD	CKdPADff
CKtPAD	CKtPADff
HDPAD	HDPADff
HDiPAD	HDiPADff
HDdPAD	HDdPADff
INPAD	INPADff
BIPAD	BIPADff
BIiPAD	BIiPADff
BIorPAD	BIorPADff

B.27.1.3.1 CKPADff, CKdPADff, and CKtPADff

Figure B-25: CKPADff, CKtPADff, and CKdPADff



CKPADff, CKtPADff, and CKdPADff are pASIC3, QuickRAM, and QuickPCI-specific clock pad macros. They must be used for pins labeled `ACLK/I` or `GCLK/I` in the pin-out tables located in the device appendices. The maximum number of available clock pads depends upon the specific pASIC3, QuickRAM, and QuickPCI device being used.

The `ACLK/I` pins can drive both the array clock network and the high-drive network. It is important to note that the array clock nets can drive the control signals of the logic cell registers (clock, set, reset), but not the input/feedback registers in the `I/O` cells. Consider the `ACLK/I`

© 2008 QuickLogic Corporation

pins as directly driving only the Array of logic cells. Alternatively, the global clock nets can drive the register control signals in both the logic cells and I/O cells as well as the OEs. The `GCLK/I` pins can drive both the global clock network and the high-drive network, and are therefore distinguished as Global pins.

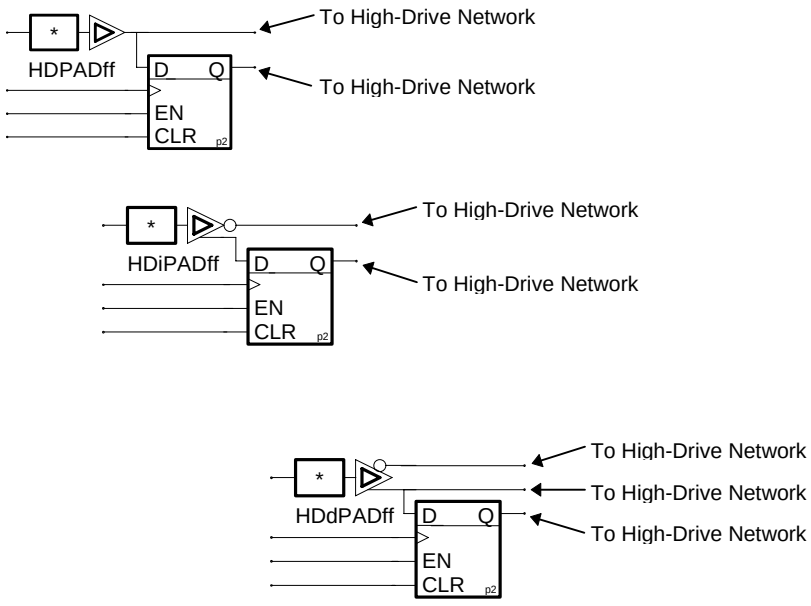
The CKPADff macro, with the addition of an input register, is the pASIC3, QuickRAM, or QuickPCI-specific version of the generic CKPAD macro. Please note that the input register is driven by the non-inverted high-drive output of the pin, and therefore, can not drive the Array Clock network directly. But the high-drive output can be connected to the Global Clock network via the `GCLKBUFF` macro.

The `CKdPADff` macro provides the identical functionality of the `HDdPADff` macro. The `CKdPADff` macro must be used on the `ACLK/I` or `GCLK/I` pins while `HDdPADff` must be used on `I` pins only. The `CKdPADff` macro has the true and complement outputs which can only drive the high-drive network. The high-drive network utilizes the quad, hex, and express routing resources to handle high fan-out nets.

The `CKtPADff` macro provides similar functionality to the `CKtPAD` macro. The `CKtPADff` macro has three non-registered outputs and one registered output. The center output from the pad can only drive the array clock or global clock networks. The outer pair of outputs can feed the true and complement signals to the high-drive nets. The input register which is driven by the high-drive terminal also feeds the high drive network. The `CKtPADff` macro should be used for flip-flop control signals which also need to be used in other combinatorial logic functions.

B.27.1.3.2 HDPADff, HDiPADff, and HDdPADff

Figure B-26: HDPADff, HDiPADff, and HDdPADff



HDPADff, HDiPADff, and HDdPADff are high-drive pad macros which are specific to the pASIC3, QuickRAM, and QuickPCI devices. They must be used on pins labeled “I” only (not ACLK/I or GCLK/I) in the pin-out tables shown in the device appendices. The maximum number of high-drive pads available depends on the specific pASIC3, QuickRAM, and QuickPCI device being used.

© 2008 QuickLogic Corporation

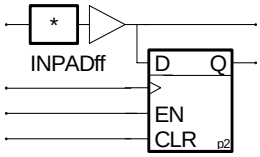
The high-drive pads are important resources in the pASIC architecture, providing twice the drive current of normal input pads (built from the bi-directional cells). High-drive pads are built from input cells.

Although the generic HDPAD, HDiPAD, and HDdPAD macros can also be used in pASIC3, QuickRAM, and QuickPCI designs, the designer can explicitly implement the inputs registers using the pASIC3, QuickRAM, and QuickPCI specific macros. `HDPADff` and `HDdPADff` are functionally similar to the generic high-drive macros with the exception of the input registers. The `HDPADff` macro provides a non-inverting output while the `HDdPADff` macro provides outputs with both polarities.

The register's control signals can be driven by the global clock network or by any internal routing resource (this excludes the array clock network driven by the `ACLK/I` pins).

B.27.1.3.3 INPADff

Figure B-27: INPADff

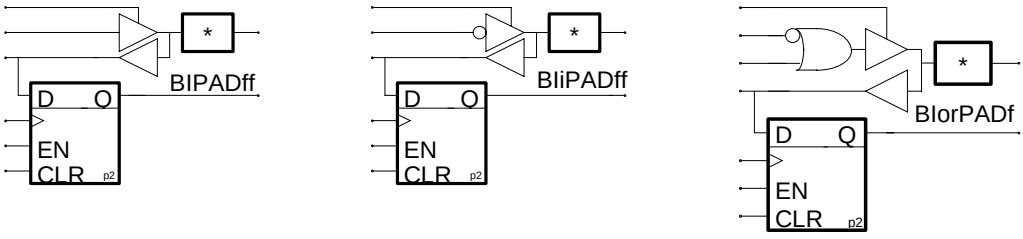


The INPADff and its multiple-instance versions are input pad macros which are specific to the pASIC3, QuickRAM, and QuickPCI devices. They must be used on pins labeled `I/O` in the pin-out tables shown in the device appendices. The `INPADff` macro provides the same non-inverting input as the generic `INPAD` macro, with the addition of the registered version of this signal. In older software versions, the bussed `INPADff` was named `INPADxff`. For compatibility that macro is located in *OLDMACROS*. For new designs, the bussed `INPADff` is named `IPADxff` and is located in the *PAD* library.

The input register's control signals can be driven by the global clock network or by any internal routing resource. The array clock network is excluded.

B.27.1.3.4 BIPADff, BIiPADff, and BIorPADff

Figure B-28: BIPADff, BIiPADff, and BIorPADf



The BIPADff, BIiPADff, and BIorPADf are bi-directional pad macros which are specific to the pASIC3, QuickRAM, and QuickPCI devices. They must be used on pins labeled `I/O` in the pin-out tables shown in the device appendices.

© 2008 QuickLogic Corporation

Although the generic BIPAD, BIiPAD, and BIorPAD macros can also be used in pASIC3, QuickRAM, and QuickPCI designs, the designer can explicitly implement the input/feedback registers using the pASIC3, QuickRAM, and QuickPCI-specific macros. `BIPADff, BIiPADff,` and `BIorPADf` are functionally equivalent to the generic bi-directional macros with the exception of the input/feedback registers. The `BIPADff` macro provides a non-inverting output while the `BIiPADff` provides an inverting output. The `BIorPADf` macro provides the full feature set of the `I/O` cell with the output OR-gate.

In older versions of the software, the bussed `BIPADff` was named `BIPADxff`. For compatibility that macro is located in `OLDMACROS`. For new designs, the bussed `BIPADff` is named `BPADxff` and is located in the `PAD` library.

The input/feedback register is driven by the input buffer of the `I/O` cell. It functions as an input register when the output enable signal is de-asserted. As the pad symbol implies, both registered and non-registered signals from the input buffer are available to the internal array of routing resources. The register functions as a feedback register when the output-enable signal is asserted. This allows a signal to drive the `I/O` pin and to be registered for internal use.

B.27.2 Master Cells

The pASIC3, QuickRAM, and QuickPCI devices have a specific set of master cells. The master cells represent the underlying building blocks of a pASIC device. They are found at the lowest level in the design hierarchy. All QuickLogic hard macros are built from the master cells. Designers can use these cells to build custom hard macros. This should be done in the Symbol Editor, by selecting **Edit>Symbol Type.** There are four pASIC3, QuickRAM, and QuickPCI master cells: the logic cell, bi-directional cell, input cell, and clock cell.

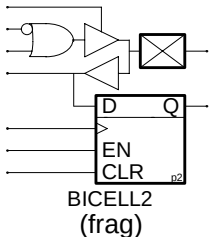
NOTE: The hard macro's symbol needs to be changed to the `CELL` type.

B.27.2.1 LCELL2

The pASIC3, QuickRAM, and QuickPCI master logic cell, LCELL2 consists of two 6-input AND gates, four 2-input AND gates, three 2-to-1 multiplexers, a D-type flip-flop, and 2-to-1 multiplexers. This increases the total number of inputs into the logic cell and allows the 6-input AND fragments to be truly independent.

B.27.2.2 BICELL2

Figure B-29: BICELL2

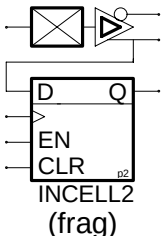


© 2008 QuickLogic Corporation

BICELL2 is used to implement the bi-directional pad macros with registers: BIPADff and its multiple-instance versions, BIiPADff, and BIorPADf; and input pad macros with registers: INPADff and its multiple-instance versions. The `BIorPADf` hard macro offers the full capabilities of this master cell, including the `OR` gate. There is no incentive to use the `BICELL2` directly.

B.27.2.3 INCELL2

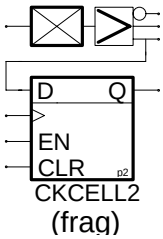
Figure B-30: INCELL2



`INCELL2` implements the high-drive macros: `HDPADff` and `HDdPADff`. The `HDdPADff` macro offers the full capabilities of this master cell. There is no incentive to use `INCELL2` directly.

B.27.2.4 CKCELL2

Figure B-31: CKCELL2



`CKCELL2` is used to implement the clock input macros, `CKdPADff` and `CKtPADff`. The `CKCELL2` cell provides access to the array clock network or the global clock network. The `CKtPADff` macro offers the full capabilities of this master cell. There is no incentive to use `CKCELL2` directly.

© 2008 QuickLogic Corporation

B.28 QuickPCI-Specific Macros

The QuickLogic Macro Library contains a set of QuickPCI-specific macros besides the generic ones mentioned earlier in this chapter. The QuickPCI macros allow the explicit implementation of QuickPCI architectural features such as: the array clock networks, global clock networks, and PCI core interface in schematic-based designs.

If specific QuickPCI features are not required, then generic macros should be used. Designs using generic macros can be targeted for either a pASIC3, QuickRAM, QuickPCI, and Eclipse device. However, the QuickPCI-specific macros are not supported for pASIC3, QuickRAM, and Eclipse designs.

Table B-139 contains all of the QuickPCI-specific macros and the section where they are described.

Table B-139: QuickPCI-Specific Macros

QuickPCI-Specific Macro Name	Library Reference Section
PCI32_25um (This macro can only be used in QuickPCI - QL5632 and QL5732 devices)	PCI Interface Macros
PCI32TV2 (This macro can only be used in QuickPCI - QL5810, QL5820 and QL5840 devices)	PCI Interface Macros
PCI32V2 (This macro can only be used in QuickPCI - QL5822 and QL5842 devices)	PCI Interface Macros

© 2008 QuickLogic Corporation

B.29 Eclipse-Specific Macros

The *QuickLogic Macro Library* contains a set of Eclipse specific macros besides the generic ones mentioned earlier in this chapter. The Eclipse macros allow the explicit implementation of Eclipse architectural features such as: the input/feedback registers, array clock networks, and global clock networks in schematic-based designs. If specific Eclipse features are not required, the generic macros should be used. Designs using generic macros can be targeted for either a pASIC3, QuickRAM, QuickPCI, or Eclipse device. However, the Eclipse and EclipsePlus-specific macros are not supported for pASIC3, QuickRAM, and QuickPCI designs.

NOTE: For Verilog or VHDL designs, please refer to the Placer chapter for discussions on pad type implementations.

Table B-140 contains all of the Eclipse and EclipsePlus (0.25 um devices) specific macros and the section where they are described.

Table B-140: Eclipse Macro Names

Eclipse-Specific Macro Name	Library Reference Name
ACCUM8_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
ACCUM16_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
ADD8_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
ADD8REG_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
ADD16_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
ADD16REG_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
BIPAD_25um	
BIPADeioff_25um	
BIPADiff_25um	
BIPADioff_25um	
BIPADod_25um	
BIPADoff_25um	
BIPADos_25um	
BPAD4_25um	
BPAD4iff_25um	
BPAD8_25um	
BPAD8iff_25um	
BPAD16_25um	
BPAD16iff_25um	
CKPAD_25um	
EIO_25um	

© 2008 QuickLogic Corporation

Table B-140: Eclipse Macro Names (Continued)

Eclipse-Specific Macro Name	Library Reference Name
GCLKBUFF_25um	
HDPAD_25um	
INPAD_25um	
INPADff_25um	
IO_BUFF_25um	
IPAD4_25um	
IPAD4ff_25um	
IPAD8_25um	
IPAD8ff_25um	
IPAD16_25um	
IPAD16ff_25um	
LVDS_INPAD_25DC	Low Voltage Differential Signal (LVDS)
LVDS_OUTPAD_25DC	Low Voltage Differential Signal (LVDS)
LVPECL_INPAD_25	Low Voltage Positive Emitter Coupled Logic (LVPECL)
LVPECL_OUTPAD_25AC	Low Voltage Positive Emitter Coupled Logic (LVPECL)
LVPECL_OUTPAD_25DC	Low Voltage Positive Emitter Coupled Logic (LVPECL)
MAC16_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
MULT_ADD16_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
MULT_ADD16REG_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
MULT8x8_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
MULT8x8REG_25um (This macro can only be used in EclipsePlus designs)	Embedded Computational Units (ECUs)
OUTPAD_25um	
OUTPADff_25um	
OPAD4_25um	
OPAD4ff_25um	
OPAD8_25um	
OPAD8ff_25um	
OPAD16_25um	
OPAD16ff_25um	
PLL_CLKPAD_25um	
PLL_DIV2HF	

© 2008 QuickLogic Corporation

Table B-140: Eclipse Macro Names (Continued)

Eclipse-Specific Macro Name	Library Reference Name
PLL_DIV2LF	
PLL_DIV4	
PLL_HF	
PLL_LF	
PLL_MULT2HF	
PLL_MULT2LF	
PLL_MULT4	
PLL_OUTPAD_25um	
PLL_RSTPAD_25um	
PLL_TOP	
RAM128x18_25um	RAM Blocks for 0.25um Devices
RAM256x9_25um	RAM Blocks for 0.25um Devices
RAM512x4_25um	RAM Blocks for 0.25um Devices
RAM1024x2_25um	RAM Blocks for 0.25um Devices
REG_LVDS_OUTPAD_25DC	Low Voltage Differential Signal (LVDS)
REG_LVPECL_OUTPAD_25AC	Low Voltage Positive Emitter Coupled Logic (LVPECL)
REG_LVPECL_OUTPAD_25DC	Low Voltage Positive Emitter Coupled Logic (LVPECL)
RG4_25um	Registers for 0.25um Devices
RG8_25um	Registers for 0.25um Devices
RG16_25um	Registers for 0.25um Devices
RGC4_25um	Registers for 0.25um Devices
RGC8_25um	Registers for 0.25um Devices
RGC16_25um	Registers for 0.25um Devices
RGE4_25um	Registers for 0.25um Devices
RGE8_25um	Registers for 0.25um Devices
RGE16_25um	Registers for 0.25um Devices
RGEC4_25um	Registers for 0.25um Devices
RGEC8_25um	Registers for 0.25um Devices
RGEC16_25um	Registers for 0.25um Devices
TRIPAD_25um	
TRIPADff_25um	
TRIPADod_25um	
TRIPADos_25um	
TPAD4_25um	
TPAD4ff_25um	

© 2008 QuickLogic Corporation

Table B-140: Eclipse Macro Names (Continued)

Eclipse-Specific Macro Name	Library Reference Name
TPAD8_25um	
TPAD8ff_25um	
TPAD16_25um	
TPAD16ff_25um	

B.29.1 Logic Cell Macro (SUPER_CELL)

The SUPER_CELL macro can be used to implement logic, which can be mapped directly into the logic cell. This macro is found in the MCELL symbol library. Since SUPER_CELL is a hard macro (and not a master cell), it can be used more than once (i.e. multiple times) within a particular schematic hierarchical level.

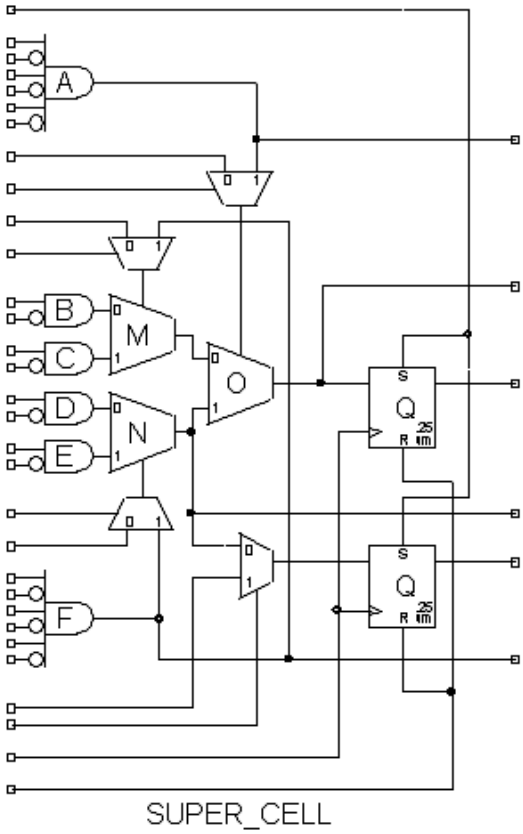
B.29.2 Master Cells for 0.25um Devices

The Eclipse devices have a specific set of macro cells similar to those of the pASIC3 family. The master cells represent the underlying building blocks of a pASIC device. They are found at the lowest level in the design hierarchy. All QuickLogic hard macros are built from the master cells. Designers can use these cells to build custom hard macros. This should be done in the Symbol Editor, by selecting Edit>Symbol Type. There are four Eclipse and EclipsePlus master cells: the logic cell, bi-directional cell, input cell, and clock cell.

NOTE: The hard macros' symbols need to be changed to the CELL type.

© 2008 QuickLogic Corporation

B.29.2.1 SUPER_CELL

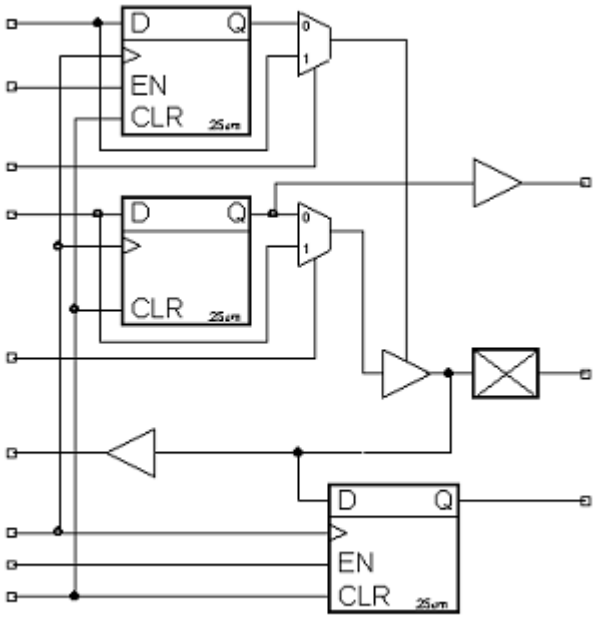


SUPER_CELL

The Eclipse master logic cell, called SUPER_CELL consists of two 6-input AND gates, four 2-input AND gates, six 2-to-1 multiplexers, and two D-type flip-flops.

© 2008 QuickLogic Corporation

B.29.2.2 EIO_CELL



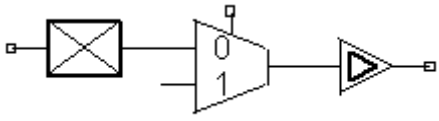
EIO_Cell

The master bi-directional cell, EIO_CELL is used when the input/feedback register is required for Eclipse designs. EIO_CELL is used to implement:

- Bi-directional pad macros: BIPAD_25um, BPAD4_25um, BPAD8_25um, BPAD16_25um, BIPADeioff_25um, BIPADiff_25um, BPAD4iff_25um, BPAD8iff_25um, BPAD16iff_25um, BIPADioff_25um, BIPADod_25um, BIPADoff_25um, and BIPADos_25um.
- Input pad macros: INPAD_25um, IPAD4_25um, IPAD8_25um, IPAD16_25um, INPADff_25um, IPAD4ff_25um, IPAD8ff_25um, and IPAD16ff_25um.
- Output pad macros: OUTPAD_25um, OPAD4_25um, OPAD8_25um, OPAD16_25um, OUTPADff_25um, OPAD4ff_25um, OPAD8ff_25um, and OPAD16ff_25um.
- Tri-state pad macros: TRIPAD_25um, TPAD4_25um, TPAD8_25um, TPAD16_25um, TRIPADff_25um, TPAD4ff_25um, TPAD8ff_25um, TPAD16ff_25um, TRIPADod_25um, and TRIPADos_25um.

NOTE: There is no incentive to use EIO_CELL directly.

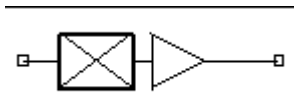
B.29.2.3 INCELL_25um



© 2008 QuickLogic Corporation

The master input cell, INCELL_25um is used when the input is required for Eclipse designs. INCELL_25um is used to implement the high-drive macro: HDPAD_25um. There is no incentive to use INCELL_25um directly.

B.29.2.4 CKCELL_25um



The master clock cell, CKCELL_25um is used when the clock is required for Eclipse designs. CKCELL_25um is used to implement the clock input macro, CKPAD_25um. The CKPAD_25um cell provides access to the array clock network or the global clock network. There is no incentive to use CKCELL_25um directly.

© 2008 QuickLogic Corporation

B.30 PolarPro-Specific Macros

The *QuickLogic Macro Library* contains a set of PolarPro specific macros besides the generic ones mentioned earlier in this chapter. The PolarPro macros allow the explicit implementation of PolarPro architectural features such as the double data rate IOs. Designs using generic macros can be targeted for either a pASIC3, QuickRAM, QuickPCI, Eclipse or PolarPro device. However, the PolarPro-specific macros are not supported for pASIC3, QuickRAM, QuickPCI and Eclipse designs.

Table B-141 contains all of the PolarPro specific macros and the section where they are described.

Table B-141: PolarPro Macro Names

PolarPro-Specific Macro Name	Library Reference Name
Ddr_dq	I/O Pads
Ddr_dqs	I/O Pads
Ddr_4dq	I/O Pads
Ddr_8dq	I/O Pads
Ddr_16dq	I/O Pads
Ddr_diffP_clkin	I/O Pads
Ddr_diff_clkout	I/O Pads
Ram4k	RAM Blocks for PolarPro Devices
Ram_8k_hc	RAM Blocks for PolarPro Devices
Ram_8k_vc	RAM Blocks for PolarPro Devices
Ram_8k_vc_dp	RAM Blocks for PolarPro Devices
ccm_ded_in_ded_fed	CCM Block

B.30.1 Logic Cell Macro (logic_cell)

The logic_cell macro can be used to implement logic, which can be mapped directly into the logic cell. This macro is found in the MCELL symbol library. Since logic_cell is a hard macro (not a master cell), it can be used multiple times within a particular schematic hierarchical level.

The logic cell in PolarPro II architecture is the same as that of the PolarPro architecture.

B.30.2 Master Cells for PolarPro Devices

The PolarPro devices have a specific set of macro cells similar to those of the Eclipse family. The master cells represent the underlying building blocks of a pASIC device. They are found at the lowest level in the design hierarchy. All QuickLogic hard macros are built from the master cells. Designers can use these cells to build custom hard macros using the Symbol Editor (select **Edit>Symbol Type**). There are six PolarPro master cells:

- logic_cell_macro
- gpio_cell_macro

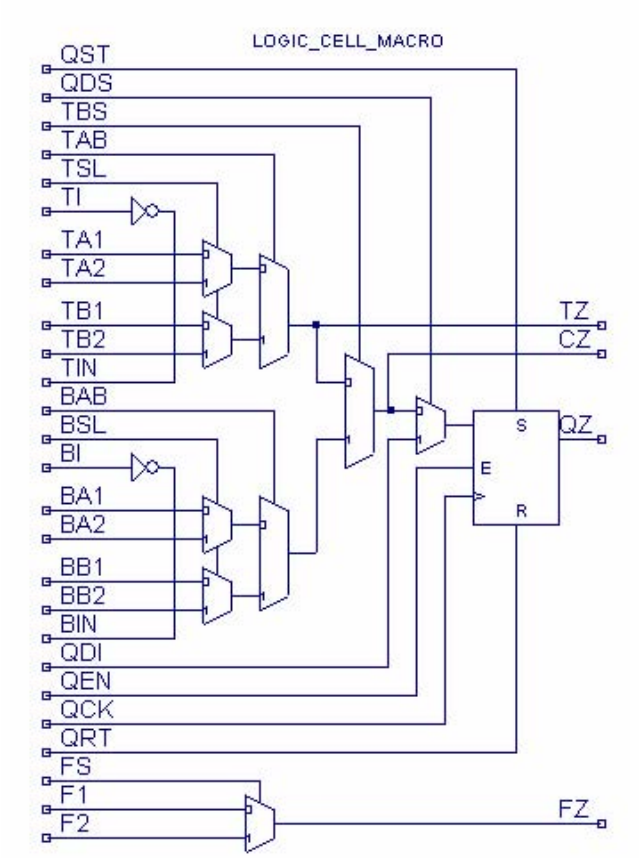
© 2008 QuickLogic Corporation

- ddrio_dqs_cell_macro
- ddrio_dq_cell_macro
- ccm_cell_macro
- ram4k_2x1_cell_macro

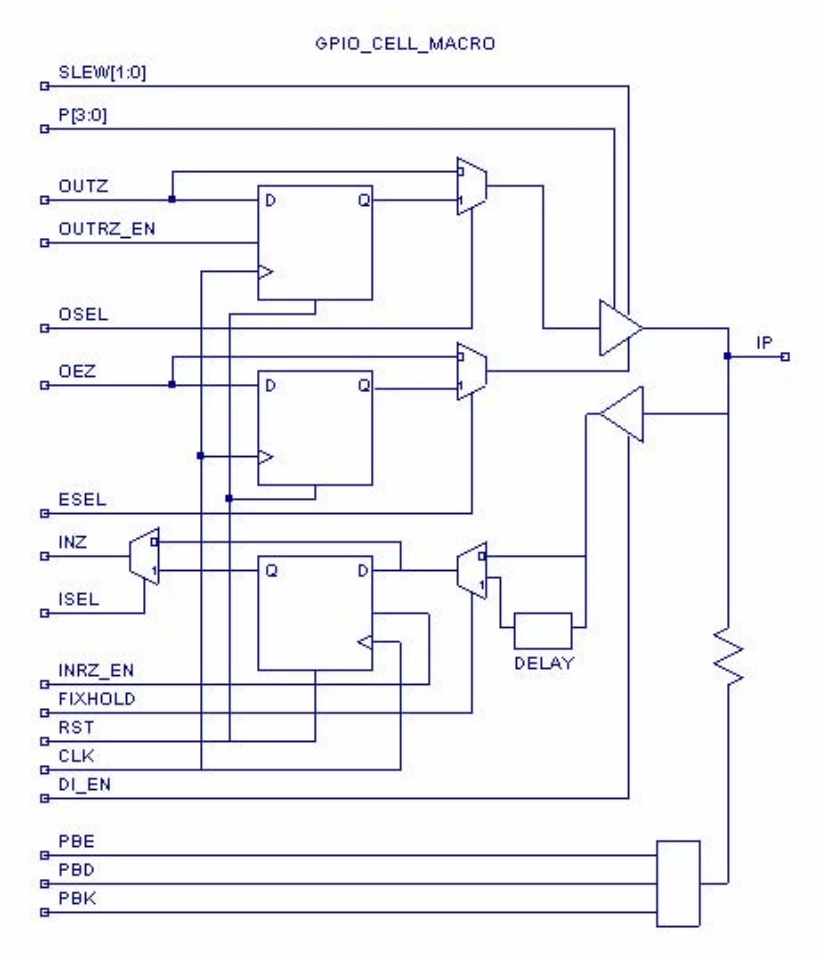
NOTE: The hard macros' symbols need to be changed to the CELL type.

B.30.2.1 LOGIC_CELL_MACRO

The PolarPro logic cell, called LOGIC_CELL_MACRO, is different from the Eclipse logic cell. The PolarPro logic cell consists of nine 2-to-1 multiplexers, two invertors, and a D-type flip-flop.



B.30.2.2 GPIO_CELL_MACRO



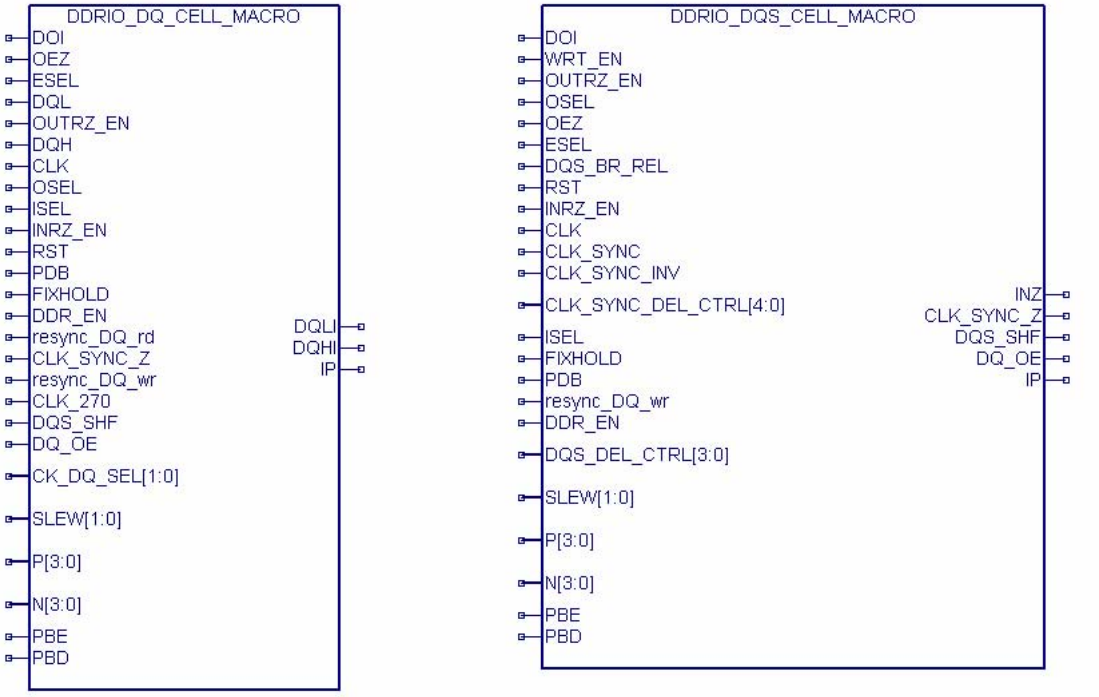
The GPIO_CELL_MACRO is used to implement the following:

- Bi-directional pad macros: bipad, bipadeioff, bipadff, bipadiff, bipadioff, bipadod, bipadoff, bipados, bpad16, bpad16ff, bpad16iff, bpad4, bpad4ff, bpad4iff, bpad8, bpad8ff, bpad8iff
- Input pad macros: inpad, inpadff, ipad16, ipad16ff, ipad4, ipad4ff, ipad8, ipad8ff
- Output pad macros: outpad, outpadff, opad16, opad16ff, opad4, opad4ff, opad8, opad8ff
- Tri-state pad macros: tripad, tripadff, tripadod, tripados, tpad16ff, tpad4, tpad4ff, tpad8, tpad8ff

NOTE: There is no incentive to use GPIO_CELL_MACRO directly.

© 2008 QuickLogic Corporation

B.30.2.3 DDRIO_DQ_CELL_MACRO, DDRIO_DQS_CELL_MACRO

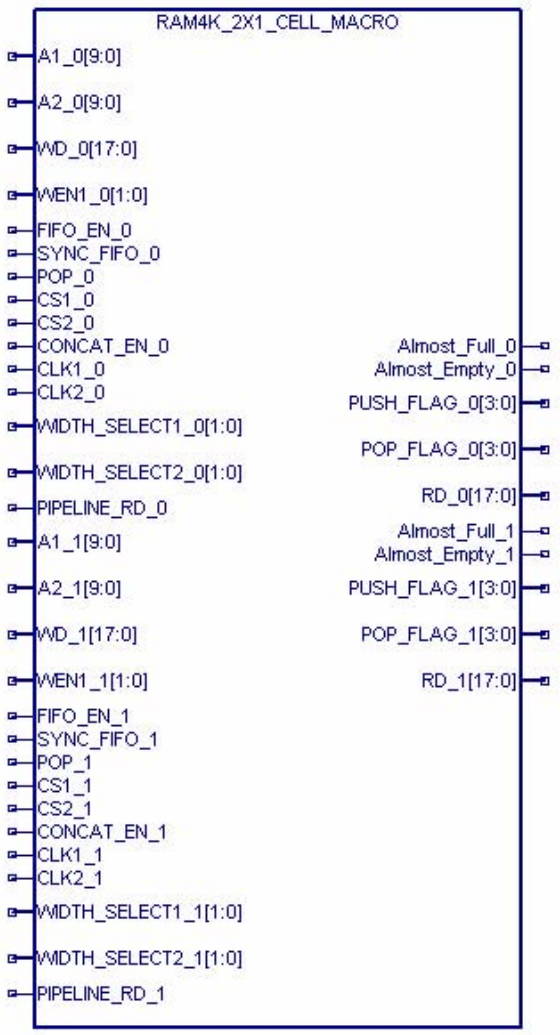


The DDRIO_DQ/DQS_CELL_MACRO are used to get the input/output data at double the clock frequency. These cells are used to implement the following macros: ddr_4dq, ddr_8dq, ddr_16dq, ddr_diff_clk_in and ddr_diff_clk_out.

NOTE: There is no incentive to use these cells directly.

© 2008 QuickLogic Corporation

B.30.2.4 RAM4K_2X1_CELL_MACRO



The RAM4K_2X1_CELL_MACRO is used to create different configurations of the RAM in PolarPro devices. This is used to implement the following macros: ram4k, ram8k_hc, ram8k_vc, ram8k_vc_dp.

NOTE: There is no incentive to use this cell directly.

© 2008 QuickLogic Corporation

B.31 PolarPro II-Specific Macros

The *QuickLogic Macro Library* contains a set of PolarPro II-specific macros besides the generic ones mentioned earlier in this chapter. The PolarPro II macros allow the explicit implementation of PolarPro II architectural features such as the 2 K and 4 K RAM resources. Designs using generic macros can be targeted for either a pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro, or PolarPro II device. However, the PolarPro II-specific macros are not supported for pASIC3, QuickRAM, QuickPCI, Eclipse and PolarPro designs.

Table B-142 contains all of the PolarPro specific macros and the section where they are described.

Table B-142: PolarPro II Macro Names

PolarPro II-Specific Macro Name	Library Reference Name
QHSCK	I/O Pads
QHSCKBUFF	I/O Pads
QHSCKBUFF2	I/O Pads
QHSCKIBUFF	I/O Pads
Ram2k	RAM Blocks for PolarPro II Devices
Ram_4k_hc	RAM Blocks for PolarPro II Devices
Ram_4k_vc	RAM Blocks for PolarPro II Devices
Ram_4k_vc_dp	RAM Blocks for PolarPro II Devices
Ram4k	RAM Blocks for PolarPro II Devices
Ram_8k_hc	RAM Blocks for PolarPro II Devices
Ram_8k_vc	RAM Blocks for PolarPro II Devices
Ram_8k_vc_dp	RAM Blocks for PolarPro II Devices

B.31.1 Master Cells for PolarPro II Devices

The PolarPro II devices have a specific set of macro cells similar to those of the Eclipse family. The master cells represent the underlying building blocks of a pASIC device. They are found at the lowest level in the design hierarchy. All QuickLogic hard macros are built from the master cells. Designers can use these cells to build custom hard macros using the Symbol Editor (select **Edit>Symbol Type**). There are five PolarPro II master cells:

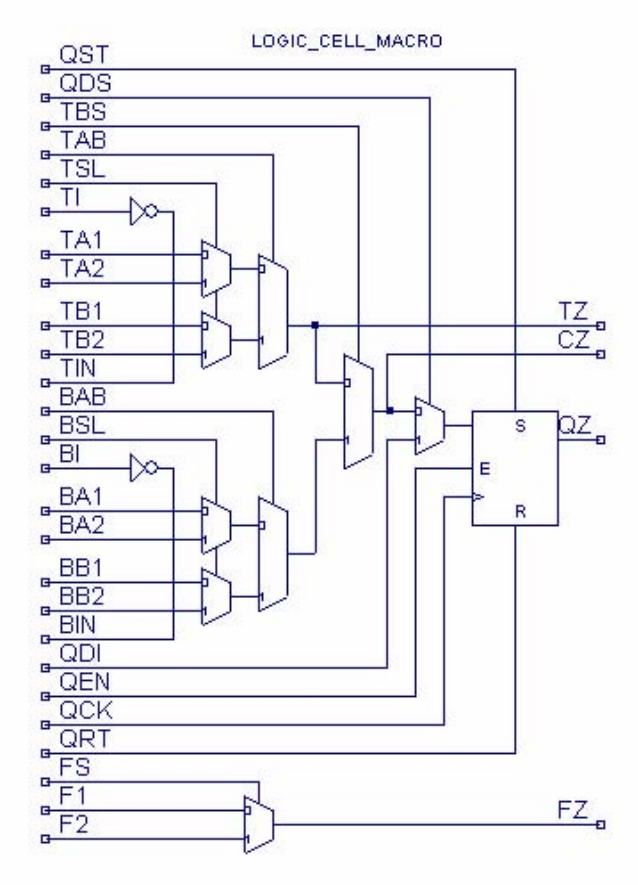
- logic_cell_macro
- gpio_cell_macro
- ccm_cell_macro
- ram2k_2x1_cell_macro
- ram4k_2x1_cell_macro

NOTE: The hard macros' symbols need to be changed to the CELL type.

© 2008 QuickLogic Corporation

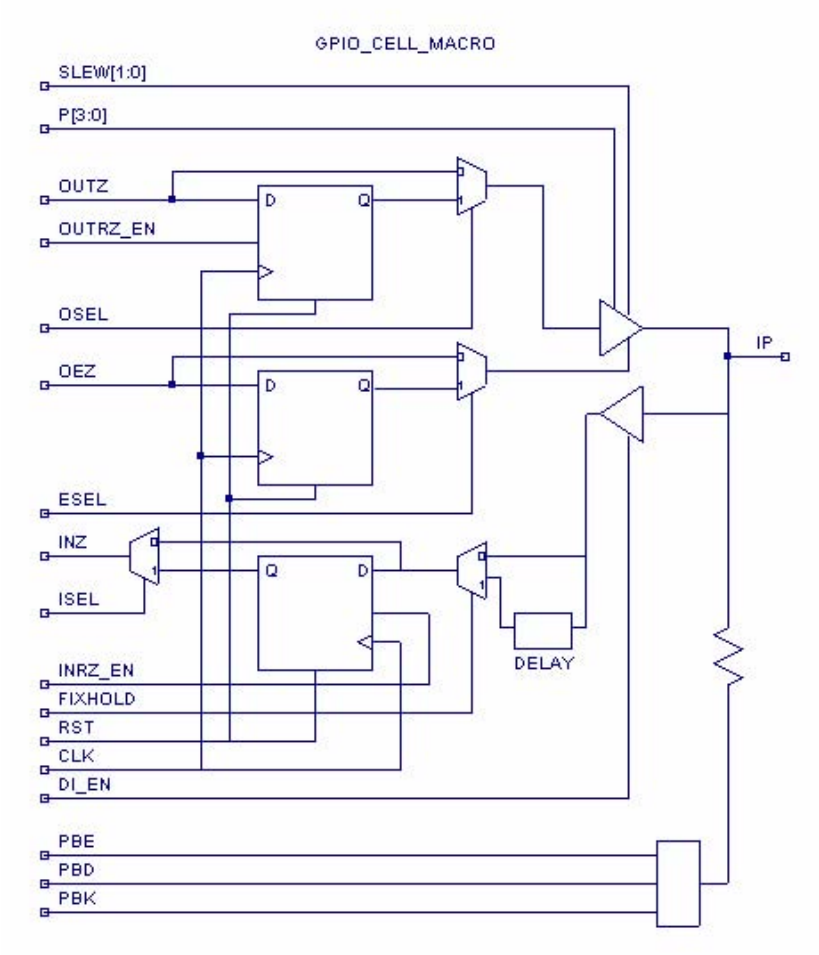
B.31.1.1 LOGIC_CELL_MACRO

The PolarPro II logic cell, called LOGIC_CELL_MACRO, is the same as from the PolarPro family.



© 2008 QuickLogic Corporation

B.31.1.2 GPIO_CELL_MACRO



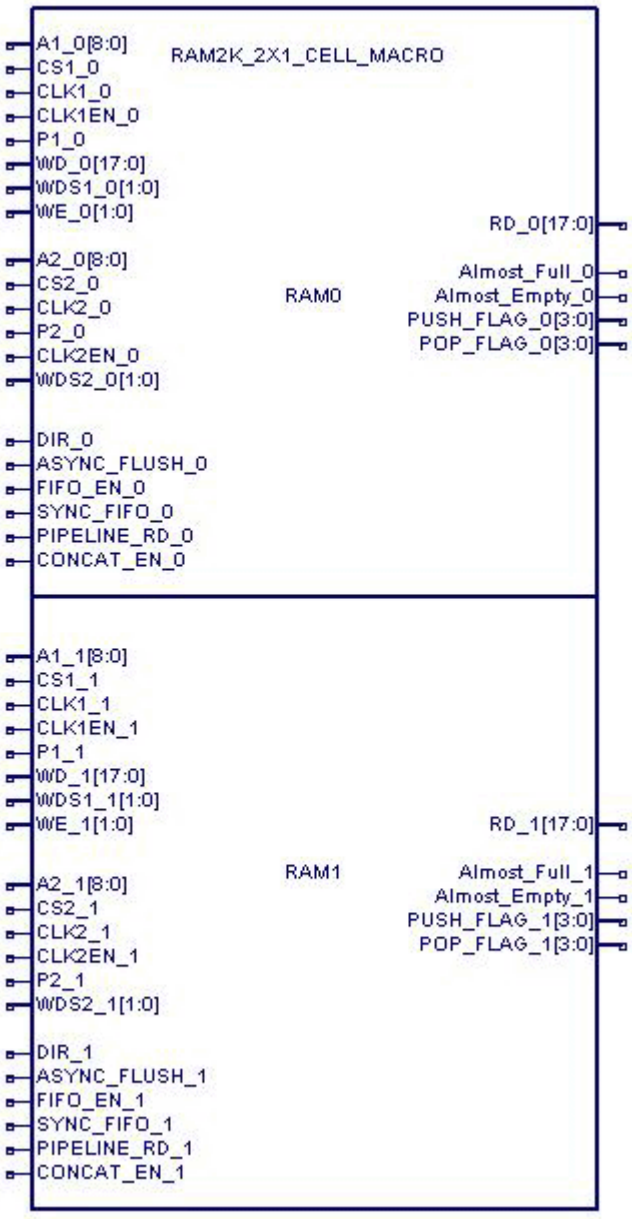
The GPIO_CELL_MACRO is used to implement the following:

- Bi-directional pad macros: bipad, bipadeioff, bipadff, bipadiff, bipadioff, bipadod, bipadoff, bipados, bpad16, bpad16ff, bpad16iff, bpad4, bpad4ff, bpad4iff, bpad8, bpad8ff, bpad8iff
- Input pad macros: inpad, inpadff, ipad16, ipad16ff, ipad4, ipad4ff, ipad8, ipad8ff
- Output pad macros: outpad, outpadff, opad16, opad16ff, opad4, opad4ff, opad8, opad8ff
- Tri-state pad macros: tripad, tripadff, tripadod, tripados, tpad16ff, tpad4, tpad4ff, tpad8, tpad8ff

NOTE: There is no incentive to use GPIO_CELL_MACRO directly.

© 2008 QuickLogic Corporation

B.31.1.3 RAM2K_2X1_CELL_MACRO

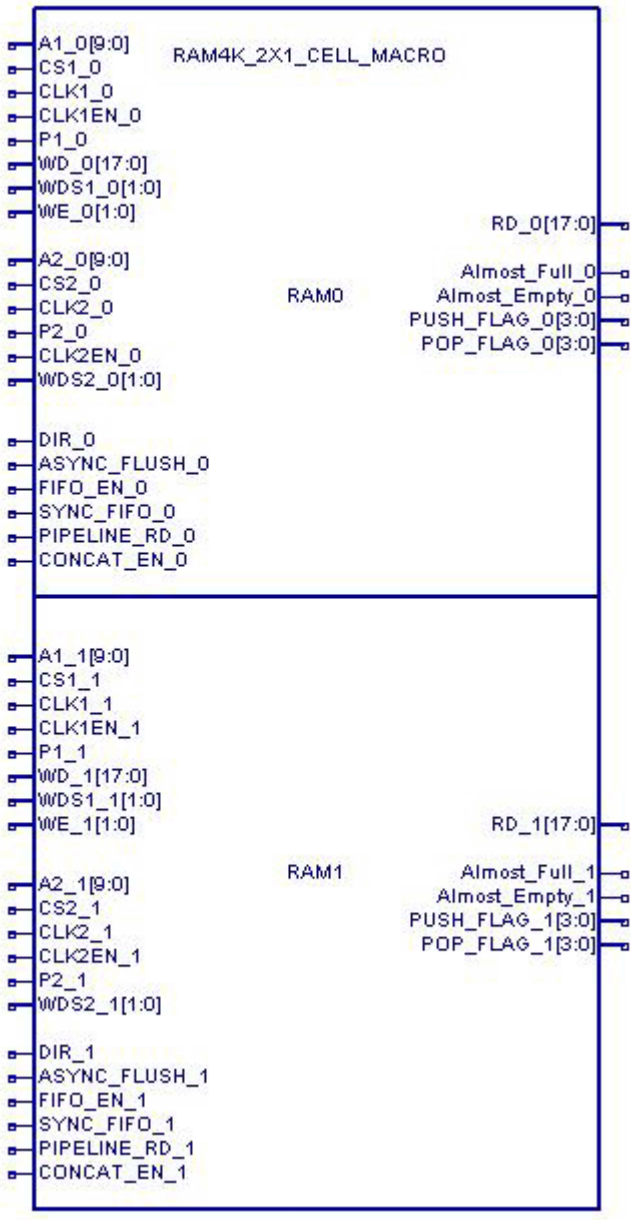


The RAM2K_2X1_CELL_MACRO is used to create different configurations of the RAM in PolarPro II devices. This is used to implement the following macros: ram2k, ram4k_hc, ram4k_vc, and ram4k_vc_dp.

NOTE: There is no incentive to use this cell directly.

© 2008 QuickLogic Corporation

B.31.1.4 RAM4K_2X1_CELL_MACRO



The RAMKK_2X1_CELL_MACRO is used to create different configurations of the RAM in PolarPro II devices. This is used to implement the following macros: ram4k, ram8k_hc, ram8k_vc, and ram8k_vc_dp.

NOTE: There is no incentive to use this cell directly.

© 2008 QuickLogic Corporation

Appendix C
QDIF File Format

The QuickLogic Data Interchange Format (QDIF) is fully specified and implemented. QDIF provides open access to the framework of SpDE for additional user or third-party tools. SpDE has a single ASCII file format, which allows for read and write interface to other systems. Only the netlist portion of QDIF needs to be created if SpDE is going to be used for place and route.

This appendix contains the following sections:

- **"QDIF Netlist Sections" on page 523**
- **"QDIF Example" on page 525**

C.1 QDIF Netlist Sections

C.1.1 Library and Logic

The Library and Logical sections are needed for SpDE to read a netlist.

Library section—the library describes how each gate type used in the netlist can be implemented in a pASIC macro cell. For example, if a two input `AND` gate is used in several places in the user's design, the implementation of an `AND` gate in the pASIC macro cell is described in the library once.

Logical section—the logical section describes each instance of gates in the library section, and how these gates are interconnected. For example, if there are three `AND` gates in the user's design, each one along with information about the nets connected to them, will be described in the logical section.

C.1.2 Syntax

The netlist sections describe a design in terms of the following:

Gate: an instance of a logic gate, such as an `AND` gate or multiplexer. Gates are often referred to as *symbols* or *instances* in schematics.

Term: a terminal on a gate. Terminals are connection points on gates to nets. Terminals are often referred to as *pins* on *symbols* in schematics.

Net: nets interconnect gates. Nets are often referred to as *wires* in schematics.

Cell: cells in the pASIC part. Cells are locations in the physical part where gates can be implemented. Types of cells include `BIDIR`, `INPUT`, `CLOCK` pad cells, `LOGIC`, `RAM`, `ECU` and `PLL` cells, which fill the interior of the chip.

Port: ports are the inputs and outputs of cells in the physical chip. The library section describes gate types in terms of an implementation in one of the chip's cells. This is done by defining which ports are associated with which terms of the gate.

© 2008 QuickLogic Corporation

C.1.3 Identifiers

The Gate, Term, Net, Instance names etc. are identifiers and must begin with an alpha character. This may be followed by any printable ASCII characters. In the ASCII characters table, this ranges from the `'!'` character (ASCII 0X21) to the `'~'` character (ASCII OX7E). Space, tab and newline are token separators. Identifiers are case sensitive and limited to 99 characters in length. A comment character `'#'` may be used, and all input from the comment character to newline is ignored.

C.1.4 Keywords

The syntax has keywords to give it structure. Keywords may be in either case. However, the requirement here is somewhat different from a programming language where user identifiers cannot conflict with keywords. QuickLogic cannot restrict user choice of names for design objects such as gates, terminals, nets, etc. To meet this need, any keyword is acceptable as an object name. The syntax is carefully designed so there are no ambiguities.

C.1.5 Supply Nets

Supply nets are restricted to the reserved net names `VCC` and `GND`.

C.1.6 Object Count

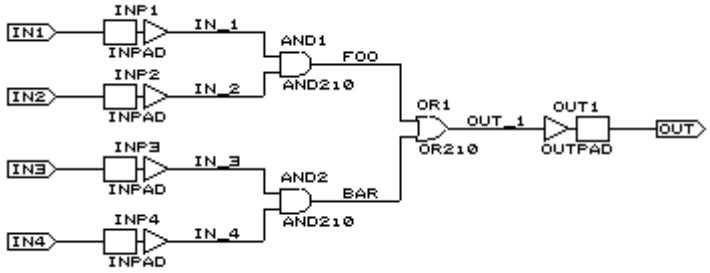
The syntax includes a count of objects before the objects themselves appear. This allows readers to operate efficiently in a single pass. Writers may have to traverse data twice in order to supply this information.

© 2008 QuickLogic Corporation

C.2 QDIF Example

Figure C-1 on page 525 is a simple circuit to illustrate a post-layout QDIF file. Instance names are shown above each macro, and net names are shown above each net.

Figure C-1: Circuit for QDIF Example




```
# This line tells the parser which version of syntax to use.
QDIF 4
# This line tells SpDE which part to use.
file QL3004
# This line tells the parser which package the part is in.
# If it is left out, a default package will be used.
package pf100
# The tools section describes which tools have been run. The “design” tool
# represents the netlister.   3000 means compatibility with release 3.0
# of SpDE, although SpDE will read netlists as far back as version 1.0.
tools
  design 3000
end
# This is the library section, which describes types of gates
library QDIF
  # The number of gate types, terminals on all gates,
  # and ports on all gates must be pre-declared.
  gates 4
  terms 18
  ports 37
  gate OUTPAD cell BIDIR
    term VCC port I1 port IE end
    term A port I2 end
    term P port IP end
    term GND end
  end
  gate OR2I0 cell LOGIC
```

© 2008 QuickLogic Corporation

```
    term VCC port F5 port F3 port F1 port E2 port D1 port C1 port B1
             port A5 port A3 port A1 end
    term GND port F6 port E1 port D2 port C2 port B2 port A6 port A4
             port A2 end
    term A port F2 end
    term B port F4 end
    term Q port NZ end
  end
  gate AND2I0 cell LOGIC
    term VCC port A5 end
    term GND port A2 port A4 port A6 end
    term A port A1 end
    term B port A3 end
    term Q port AZ end
  end
  gate INPAD cell BIDIR
    term VCC port I2 end
    term GND port I1 port IE end
    term P port IP end
    term Q port IZ end
  end
end
# This section describes gate instances and nets in the user's design.
logical QDIF
  # Again, total net and gate counts must be pre-declared
  gates 8
  nets 14
  # Cell pre-placements are written as "cell <cell name>. Cell names for
  # pads use IO<pin number> syntax, accept for pin grid arrays, which use
  # <column letter><row number> syntax. Logic cell locations use
  # <column letter><row number> syntax as well (i.e. C10 or H2).
  gate INP1 master INPAD cell IO12 end
  gate INP2 master INPAD cell IO13 end
  gate INP3 master INPAD cell IO14 end
  gate INP4 master INPAD cell IO58 end
  gate AND1 master AND2I0 end
  gate AND2 master AND2I0 end
  gate OR1 master OR2I0 end
  gate OUT1 master OUTPAD cell IO60 end
  # Writing VCC and GND nets is optional when the user does not
```


© 2008 QuickLogic Corporation

```
# directly use them.
net VCC
end
net GND
end
net IN_1
  gate AND1 term A end
  gate INP1 term Q end
end
net IN_2
  gate AND1 term B end
  gate INP2 term Q end
end
net IN_3
  gate AND2 term A end
  gate INP3 term Q end
end
net IN_4
  gate AND2 term B end
  gate INP4 term Q end
end
net FOO
  gate OR1 term A end
  gate AND1 term Q end
end
net BAR
  gate OR1 term B end
  gate AND2 term Q end
end
net OUT_1
  gate OUT1 term A end
  gate OR1 term Q end
end
net IN1 direction input
  gate INP1 term P end
end
net IN2 direction input
  gate INP2 term P end
end
net IN3 direction input
```

© 2008 QuickLogic Corporation

```
    gate INP3 term P end
  end
  net IN4 direction input
    gate INP4 term P end
  end
  net OUT direction output
    gate OUT1 term P end
  end
end
```


© 2008 QuickLogic Corporation

Appendix D
Error Messages

This appendix is a reference for all SpDE messages and contains the following sections:

- **“Error Sources” on page 529**
- **“SpDE Design Verifier” on page 530**
- **“SpDE Tool Errors” on page 535**
- **“Internal Errors” on page 539**
- **“Out of Memory Errors” on page 540**
- **“SpDE PolarPro Design Verifier” on page 541**
- **“SpDE PolarPro Tool Verifier” on page 544**

D.1 Error Sources

Error messages may result in response to action from the following sources:

- Errors from the Hierarchy Navigator, generated after selecting **Tools>Export QuickLogic**. Refer to **“SpDE Design Verifier” on page 530**.
- Errors from SpDE, generated after selecting **File>Import**. Refer to **“SpDE Design Verifier” on page 530**.
- Errors from Precision RTL, generated after selecting **File>Import Using Precision RTL**.
- All numbered error messages from QuickLogic tools. Refer to **“SpDE Tool Errors” on page 535**.
- Error Messages from other QuickLogic-supported design tools. Refer to the supplemental documentation for the design tool.

© 2008 QuickLogic Corporation

D.2 SpDE Design Verifier

The Design Verifier runs when a design is loaded into SpDE. It displays notes in the **Verifier Notes** tab of the Transcript window, and warnings and errors in the **Errors and Warnings** tab of the Transcript window.

D.2.1 Notes

Notes bring a situation to the designer's attention. The situation is usually not a problem, but should be verified nevertheless.

NOTE: Gate <> is not used, and is being removed
This message will occur for any gate that does not have its output tied to any other logic. This tends to have a ripple effect. Once a gate is removed, this can also cause the gate driving it to be removed. This can eliminate all the gates in a design in some cases. The solution is to make sure that you use all the gates in your design, and attach output logic to output pads (OUTPADs). This note can be ignored if you expect logic to be removed (such as when a 16 bit counter is used in the design, but only 13 bits are used).

D.2.2 Warnings

Warnings serve to alert the designer to a problematic situation, commonly associated with a real problem.

Table D-1: Warning Messages

WARNING: Gate <> cannot have a fixed placement
The only types of gates which can have a fixed placement are RAM blocks, flip-flop cells, and I/O cells.
WARNING: Incompatible inputs moved from Net <>
This error occurs when a net driven by the clock network driver of a clock pad is also driving logic.
WARNING: Net <> drives no inputs
The net specified has a gate that drives it, but no destinations. If this is an output net, make sure that you attach it to an output pad (OUTPAD).
WARNING: Net <> should not have arrival set <>
This message indicates that an arrival time attribute has been added to a net which was not an input. Arrival time attributes are appropriate only for input nets.
WARNING: Net <> should not have departure set <>
This message indicates that a departure time attribute has been added to a net which was not an output. Departure time attributes are appropriate only for output nets.
WARNING: Some nets have high-fanout. Auto-buffering should be enabled (currently disabled)
The auto-buffering must be enabled due to some nets with high fanout.
WARNING: Net <> connects two pads externally
Single net connected to two I/O pins.
WARNING: Global clock buffers cannot have fixed placement, the fixed placement for gate <> is ignored
Global clock buffers cannot be fix-placed by the user.

© 2008 QuickLogic Corporation

D.2.3 Errors

Errors flag genuine error conditions that would prevent parts from being programmed. However, the tools can still be run for experimental purposes and examination.

Table D-2: Error Messages

ERROR: Gate <> has floating input
The gate with the instance name specified has one or more inputs, which are not driven from any other logic. This may be a case where a pin has been left without a connection in a schematic. Tie all inputs to VCC or GND to avoid this error.
ERROR: Net <> driven by multiple I/O pads
It is illegal to drive a net from one or more I/O pads. If this was not done intentionally, look carefully at the net names attached to your input and output pads, to make sure that none of the pads are accidentally tied together with the same net.
ERROR: Net <> has fanout of < 24, but > 2 drivers
Due to the strong internal wire current when you use more than 2 high-drive pads in parallel, you cannot drive less than 24 loads within the chip. If you want to use less than 24 loads, then downgrade your driver to an HD2PAD (2 high-drive pads in parallel).
ERROR: Net <> has no driver
There is a net in your design that is not driven by a gate. This may be because you have forgotten to put input pads on your inputs. This may also have happened if you are not yet done with your design. Tie all unused inputs to VCC or GND to avoid this error.
ERROR: Missing enable net on terminal <>
If this error occurs, please contact QuickLogic Technical Hotline.

D.2.4 Fatal Errors

Fatal errors flag serious error conditions which will prevent the tools from being run.

Table D-3: Fatal Error Messages

FATAL ERROR: Clock net <> has multiple drivers
The specified clock net has more than one driver, which is illegal. Only one driver per clock network is allowed.
FATAL ERROR: Gate type <> has illegally connected OS port
A custom defined macro with the specified name has been created incorrectly. There are specific rules defining how to create a custom logic cell macro. Refer to the *Macro Library* chapter of this manual for details. If the gate name specified is not a custom macro, but a macro supplied by QuickLogic, contact the Technical Hotline.
FATAL ERROR: Bad Design File: Term <> on gate <>
If this error occurs, please contact the QuickLogic Technical Hotline.
FATAL ERROR: Gate <> drives VCC or GND
The output of the specified instance is connected to VCC or GND, which is illegal.
FATAL ERROR: Gate <> should be placed on a clock cell

© 2008 QuickLogic Corporation

Table D-3: Fatal Error Messages (Continued)

The instance name specified corresponds to a gate that must be placed on one of the clock pad locations. Clock pads may not be placed on high-drive (input-only) pad locations, I/O pad locations, or logic cell locations.
FATAL ERROR: Gate type <> has illegally connected NS port
A custom defined macro with the specified name has been created incorrectly. There are specific rules defining how to create a custom logic cell macro. Refer to the *Macro Library* chapter of this manual for details. If the gate name specified is not a custom macro, but a macro supplied by QuickLogic, contact the Technical Hotline.
FATAL ERROR: Gate type <> has illegally connected QD port
A custom defined macro with the specified name has been created incorrectly. There are specific rules defining how to create a custom logic cell macro. Refer to the *Macro Library* chapter of this manual for details. If the gate name specified is not a custom macro, but a macro supplied by QuickLogic, contact the Technical Hotline.
FATAL ERROR: Dual drive gate <> illegally connected
This refers to a double-buffer configuration in your design. Two gates have been tied in parallel (their inputs and outputs tied together), but there was an error in the wiring. Check your design. If you did not intend to create a double-buffer in your design, then look for the specified gate in your design and check the wiring around it.
FATAL ERROR: Gate <> has illegally connected outputs
This refers to a double-buffer configuration in your design. Two gates have been tied in parallel (their inputs and outputs tied together), but there was an error in the wiring. Check your design. If you did not intend to create a double-buffer in your design, then look for the specified gate in your design and check the wiring around it. If you intended to make a custom double-buffer, check the rules for double-buffering in the *Design Techniques* chapter of this manual.
FATAL ERROR: Terms on gate <> ill-defined
If this error occurs, please contact the QuickLogic Technical Hotline.
FATAL ERROR: The design has no gates!
The Verifier has removed all of the gates in the design. Make sure the design contains I/O pads.
FATAL ERROR: Bad Design File: Term <> on net <>
If this error occurs, please contact the QuickLogic Technical Hotline.
FATAL ERROR: Gate <> placed on incompatible cell
The placement attribute on the specified gate is not legal for this gate. Make sure none of the I/O pads in this design have logic cell placements, and also make sure none of the gates in the design have I/O placements.
FATAL ERROR: Gates <> and <> are placed on the same cell
The two instances specified have identical fixed-placement attributes. Change one or both of the fixed placement attributes so that the two gates are placed in different locations in the device.
FATAL ERROR: Gates <> and <> illegally paralleled
The two specified instances have outputs that drive the same net. Paralleled gates are allowed only if the two gates are identical, and can be placed in the same logic cell.
FATAL ERROR: High-drive net <> drives a global clock buffer
The high-drive net specified is connected to the input of a global clock buffer, which is illegal in PASIC 3 devices. A simple work around is to use the high-drive net to drive a logic buffer (BUFF), which, in turn, drives a global clock buffer.

© 2008 QuickLogic Corporation

Table D-3: Fatal Error Messages (Continued)

FATAL ERROR: High-drive net <> has opposing pads in a corner
This is a rare condition. The problem is that high drive nets have limited routing connections. High drive nets are nets, which are driven by HDPADs, CKTPADs, or double-buffers (two identical buffers or *AND* gates tied in parallel). It is illegal to use a high-drive net to drive two output pads (or their tri-state enables) in the corner of the device. Change the driver for the net so it is not a high-drive net, or move the pad position. If the driver is an HDPAD or a CKTPAD, try inserting a buffer in the design before driving the output pads.
FATAL ERROR: Net <> is driven by more than one high-drive pad in a pASIC 3 device
The specified net is connected to the outputs of two high-drive pads (input-only pads). Paralleling high-drive input pads is illegal in pASIC 3 devices.
FATAL ERROR: Net <> is driven by more than one IO control/hi-drive pad
The specified net is connected to the outputs of two IO control/high-drive pads (input-only pads). Paralleling high-drive input pads is illegal.
FATAL ERROR: Pad <> pre-placed on opposite side
If a High-Drive pad (HDPAD) is driving an output pad directly (or the tri-state enable for an output pad), that output pad cannot be placed on the opposite side of the chip as the HDPAD. This is due to a routing limitation on high-drive wires. If you must have the pinout as already defined, then try inserting a buffer on the output of the HDPAD, so the outputs are not driven directly from the high-drive pad, but from the buffer.
FATAL ERROR: Net <> uses array clock pad to drive incompatible inputs
The specified net is driven by the Array Clock input pad, and it drives combinatorial inputs to one or more logic cells. To resolve this problem, you may possibly use a Global Clock input pad, the Global Clock buffer, or a high-drive or I/O pad. Note that Global Clock nets may only feed the F1 logic input to the logic cell. See *The Router* chapter of this manual for more details regarding legal Global Clock connections.
FATAL ERROR: Net <> uses global clock pad to drive incompatible inputs
The Global Clock net specified drives logic cell inputs other than the F1 input, which is illegal in pASIC 3 devices. To resolve this problem, use the F1 input when possible, and in other cases, connect the Global Clock net to the F1 input of an F-fragment to create a buffer capable of driving any logic input.
FATAL ERROR: Net <> driven by more than two logic outputs
The specified net is driven by more than two drivers. No more than two drivers are allowed for any internal net. If this was not done intentionally, check the net connections on this net name carefully in your design to find a possible design error.
FATAL ERROR: The same net name <> cannot be used both inside and outside the chip
The specified net is connected to both the input and output of an I/O pad. Use a different net name on each side of an I/O pad.
FATAL ERROR: Pad <> must be pre-placed
This error is given because you are driving output, tri-state, or bi-directional pads directly from a high-drive net. High-drive nets are driven by HDPADs, CKTPADs, or double-buffers (two identical buffers or AND gates tied in parallel). When you drive outputs (or tri-state enables for outputs) directly with high-drive nets, you need to manually fix the placement of the output pads. Make sure that the placement is fixed so that no two output pads driven by the same high-drive net are on opposite sides of the chip (an illegal condition).
FATAL ERROR: Programming cell can not be used

© 2008 QuickLogic Corporation

Table D-3: Fatal Error Messages (Continued)

If this error occurs, please contact the QuickLogic Technical Hotline.
FATAL ERROR: Terms on net <> ill-defined
If this error occurs, please contact the QuickLogic Technical Hotline.
FATAL ERROR: Used <> bi-directional pads with <> available
The number of bi-directional I/O pads required for this design is greater than the number of pads available in this device-package combination. Either remove some pads from the design, or use a device-package combination with more I/Os.
FATAL ERROR: Used <> clock pads with <> available
The number of clock pads required for this design is greater than the number of clock pads available in this device-package combination. Either remove some clock signals from this design, use a device with more clock pads, or use high-drive or I/O pads for some of these signals.
FATAL ERROR: Used <> flip-flops with <> available
The number of flip-flops required for this design exceeds the number of available flip-flops in this device. Remove some flip-flops from the design, or use a larger device.
FATAL ERROR: Used <> input-only & clock pads with <> available
The number of input-only (high-drive) and clock pads required for this design exceeds the number of available input-only and clock pads available in this device. Either remove some of these signals from the design, or move some of the signals to I/O pads.
FATAL ERROR: Used <> global clock networks
The number of Global Clock networks required for this design exceeds the number of available Global Clock networks available in this device. Either move some of these signals to other routing resources, remove them from the design, or use a device with more Global Clock networks.
FATAL ERROR: The design does not pass the database check!
If this error occurs, please contact the QuickLogic Technical Hotline.
FATAL ERROR: Used <> RAM cells with <> available
The number of RAM cells required for this design is greater than the number of RAM cells available.
FATAL ERROR: Gate <> cannot be placed
This device can only support a maximum of 4 hidrive pads.
FATAL ERROR: Net <> has a global buffer that drives nothing
An OUTPAD is not used for outputs of flip-flops driven by the CKPAD or Global Clock Buffer on this net <>.
FATAL ERROR: Net <> driven by more than one <> output
A single net is driven by two of more signals.

© 2008 QuickLogic Corporation

D.3 SpDE Tool Errors

SpDE reports user errors in an Error dialog box and/or in the Transcript Window. These errors represent design or system errors which may be fixed by the user. The tables below are organized by tool code. The first two letters of the error code indicate the tool.

© 2008 QuickLogic Corporation

D.3.1 XX - (starting with any two letters)

xx0100 - 0199: Out of Memory
SpDE has requested more memory than Windows currently has available. Try closing other applications and re-run SpDE. If still a problem, try re-starting windows. Many memory problems can be solved by creating a larger Windows swap file (see the Virtual Memory section of the PC System Requirements appendix). Windows offers very efficient memory management—refer to the *Microsoft Windows User's Guide* for complete details.

D.3.2 DB - SpDE Database Module

DB0002: Invalid Package Type
An invalid package type has been chosen for the QuickLogic chip chosen. Refer to the QuickLogic documentation for the design entry package you are using to see valid package types.

D.3.3 CR - Clock Router

CR0001: Conflict in clock network assignments between <net name> & <net name>
In a column if number of clocks (network) used is more than available then there is conflict between clock nets. Such situation can be avoided by moving few load to some other column or quadrant.

D.3.4 ED - EDIF Netlist Reader

ED0001 - ED0002: Syntax error on line <line number>
At line <*line number*> in the EDIF file, illegal syntax has been used.
ED0004: Unexpected end of file
The EDIF file ended pre-maturely. This is usually because of an error that occurs during the creation of the EDIF file.
ED0007: Invalid symbol '<symbol>' on line <line number>
The EDIF file contains the string <*symbol*> with an invalid character on line <*line number*>.

D.3.5 LR - Logic Re-Optimizer

LR0006: Required number of logic cells: <*number of logic cells required*> Available: <*number of logic cells available*>
Number of logic cells in the targeted device is not sufficient to map the design. Select a device, which has more number of logic cells.
LR0007: Duplicate net: No enough space to duplicate net <*Net Name*>. fanout=<*Number Of FanOut*>
All Logic cells are fully packed. There is no space left to map the duplicated gate. Select a device with more Logic Cells.

© 2008 QuickLogic Corporation

D.3.6 LS - Load and Save Files

LS0001: Could not open binary file <*filename*>.
<*Filename*> specified by the user either does not exist or does not have a read attribute.
LS0003: Unknown part name <*part*>.
The part specified in the design file does exist or does not have an associated part file. Check your `WIN.INI` file to insure that the ini-path entry has been properly set.
LS0009: Could not find device files for '<device file>'
The device file mentioned in the message could not be found. It could be an installation problem. Try reinstalling Quick Works.
LS00010: double gate <*gate name*>.
The Gate name is duplicated in the net-list. Check your design and avoid duplicate names to the gates.
LS00011: Unknown package type: <package>
A package is specified in the QDIF file which SpDE does not recognize. Refer to Quick*Logic* documentation for your design entry tool for valid package types.
LS00012: double net <Net name>
The Net name is duplicated in the net-list. Check your design and avoid duplicate names to the Nets.
LS00200: <*Error*> at approximately line <*line number*>
The parsing error <*Error*> occurred while reading line <*line number*> of the QDIF file.

D.3.7 NM - Net Lister

NM0011: Property type error on device_type
Check the type of value for device_type property in design file. QuickWorks supports only strings values for these properties.
NM0014: Property type error on package_type
Check the type of value for package_type property in design file. QuickWorks supports only strings values for these properties.
NM0031: Primary I/O net specified in PLACE property '<*name*>=<*value*>' from <path name> not found.
To specify the placement property for a pad, its primary I/O net should be specified.
NM0042: Illegal cell '<*cell name*>' found in EDIF. Contains Frag cell mixed with other cell types.
QuickWorks does not support the fragment cells used with other cell types in a design. Please make sure, if a cell has fragment cell instances, it should not have instances of other cell types.
NM0043: Incompatible part: design with ECU in part with no ECUs.
Check the device selected for the design getting loaded. For a design, which uses ECUs, make sure a device that has ECU resources is selected.

© 2008 QuickLogic Corporation

D.3.8 RT - Router

RT0000: Could not complete routing **RT0002: Could not complete hi-drive routing**
The Router does not have enough resources to complete routing. In the case of hi-drive routing, refer to the Router chapter in the Special Routing Cases section. Otherwise, try re-placing after changing the placer seed.

D.3.9 SD - SDF Writer

SD0001: Cannot open file: *<filename>*
The SDF writer can not open the SDF file that it needs to create. This could be due to a full disk, or a write-protected file or directory.

D.3.10 SQ - Sequencer

SQ0000: Sequencer could not complete. Re-run Router with a different seed.
The Sequencer could not determine an order in which to program the Via-Links in the part after 5 routing attempts. Re-running the Router with a different seed will correct the problem.

D.3.11 TP - Placer

TP0003: Clock nets conflict with pll by fixed FF.
The FF fixed in specific quadrant caused the clock net conflict with the PLL driven by the same clock net. Unfix the FF.
TP0004: Failed to assign global clocks. Reduce one or more clock sources **TP0005: Failed clock nets assignment. Reduce one or more clock sources**
The clock assignment routing cannot assign clock nets (including PLL net, global clock buffer nets) into appropriate quadrants due to the limited number of clock networks in the device. Reduce the number of clock nets, PLL nets, and global clock buffer nets.
TP0006: Failed to resolve the conflict between two clock nets *<Net Name> & <Net Name>*
The two clock nets listed in the error message use the same clock network and the driven FFs of both clock nets fall in the same quadrant.Unfix the FF in the clock net or change clock network by relocating the clock to a different clock pad.
TP0007: Buffer block drives no net
A buffer was inserted by the tool, but it drives no fanout. No action is required from users.
TP0008: Failed to resolve the clock enable conflicts, use constraint manager for correction
The PolarPro II architecture has a dynamic clock enable feature in clock networks. If there are more than one of these enables used on a clock pad, flip-flops driven by different enable signals must not be placed on the same column by the placer. If this error occurs, the placer is not able to resolve clock enable load conflicts due to multiple clock domains with multiple enables. This can be due to user fix placement constraints or design connectivity (e.g., flip-flop driven by more than one clock enable (clk, reset etc.)). Resolve clock enable load conflicts by moving flip-flops into different columns using the fix placement tool.

© 2008 QuickLogic Corporation

D.3.12 UL - SpDE CLI

UL0004: Unknown tool <Tool Name> passed on command line
The tool name passed through the command line is wrong. Refer to CLI help for valid tool names. Note that if the tool name comprises of two or more words (e.g., back annotation), it should be placed in double quotes.
UL0005: HexFile not generated
Hex file generation failed while processing the data, either the .mem file has incorrect size and data width or the mapping location for `.mem` are incorrectly specified in the data file.
UL0026: Runscript failed
The TCL script could not be executed. It can be a TCL scripting related problem or it can be related to SpDE. Check the previous errors to know exact problem.
UL0032: No design has been loaded
The command line option used requires a design (`qdf/chp/edf` file) to be loaded. Use -LOAD option before this option. The sequence of commands is important as SpDE executes the options sequentially from left to right.

D.3.13 VE - SpDE Physical Viewer

VE0003: Error occurred while saving constraint file. Save As operation aborted.
SpDE detected an error while saving the constraint file `<design>.qcf`. Check for the write permissions on the directory path.

D.3.14 VG - Verilog Netlister

VG0001: File <> does not have write permission. Please change the permissions or login with a user having privilege to change the permissions.
The Verilog netlister cannot open the output file it is trying to create. This could be due to a full disk or a read-only directory.

D.3.15 VR - Verifier

VR0004: I/O control output driving incompatible port.
Signal connected to IO Control Pad can drive only clock/set/reset ports of I/O pads.

D.4 Internal Errors

These errors are indicated by the message: `Internal Error or Fatal Error` - accompanied by their usually cryptic nature. These errors should not occur—they indicate an inconsistency in SpDE's data structures. If one of these errors is encountered, record the text of the error completely and contact QuickLogic Corporation.

© 2008 QuickLogic Corporation

D.5 Out of Memory Errors

If tools cannot execute properly, or you get an error message like, `Unable to run toolname, check path`, or if you get an error message with any reference to being out of memory then you are probably running too many applications in Windows. Close any unnecessary applications and try again. After an out of memory error occurs, it is best to save your work and re-start Windows. When an out of memory error occurs, a tool may not close properly and leave Windows in a somewhat unstable state. You may be out of memory because of improper Virtual Memory Settings.

© 2008 QuickLogic Corporation

D.6 SpDE PolarPro Design Verifier

D.6.1 Warnings

Table D-4 lists the PolarPro-specific warning messages.

Table D-4: Warning Messages

WARNING: Gate <> is a Synchronous FIFO, Clock Nets <> and <> should be from same clock tree
Synchronous FIFO requires both RCLK and WCLK coming from the same clock tree.
WARNING: Gate <> is not configured as FIFO, Net <> drives signal
If a gate is not configured as a FIFO then the FIFO signals should not have any nets connected (i.e., the ports ALMOST_FULL / ALMOST_EMPTY / PUSH_FLAG / POP_FLAG should not drive any logic.
WARNING: Gate <> cannot have same Pull-Up and Pull-Down assignment, setting it to default
A gate cannot have both pull up/down set to true. If such gate is encountered, it would be set to the default setting for pull down or pull up.
WARNING: Pull down not available in <> Ignoring constraint
In PolarPro, the BR device pull down feature is disabled in DDRIOs. If a gate has the pull down feature turned ON and is pulled in DDRIO, the constraint would be ignored.
WARNING: Incompatible IO pulled together in one bank <>
This message indicates that GPIOs and DDRIOs are pulled into same bank.
WARNING: Number exceeds available limit
This message comes up in the IO configuration tool if the number of I/Os pulled in set/bank exceeds the available limit.
WARNING: Gate can't be mapped. Disable pull down feature and try again.
This message is presented during IO configuration, when a gate with the pull down feature turned ON is pulled into the DDRIO of a PolarPro BR device.
WARNING: Number if DQS exceed the available limit
This message indicates that more than one DQS is pulled into a set.
WARNING: Incompatible IO pulled in Strobe <>
This message comes up in the I/O Configuration tool, if any gate other than strobe is pulled into strobe.
WARNING: Incompatible standard <> selected for set
This message comes up when a user selects standard for the set that is not compatible with the IOs that are pulled in the set.
WARNING: DQ <> mapped to <> whereas DQS <> is mapped to <>, Ignoring constraint for DQ
If DQ gates and DQS gates are mapped to different sets, the constraint for DQ would be ignored.
WARNING: DQ <> not mapped to any set
DQS is mapped to a set, whereas DQ is not mapped to any set.
WARNING: diff gate pulled with <> in same set instead of <>...removing former constraint
If the diff clock gates, clk+ and clk- are pulled into two different sets or clk+ is pulled with another diff clock gate in the same set or vice versa then constraint for the other gate will be removed.

© 2008 QuickLogic Corporation

D.6.2 Errors

Table D-5 lists the PolarPro-specific error messages.

Table D-5: Error Messages

ERROR: RAM. <> cannot use more than <> address lines
Check whether the correct number of address lines are used for the given depth of the RAM.
ERROR: VCCIO of <> is set to <>, check the VCCIO for other gates in <>
All the gates pulled in a set should be set to the same VCCIO.
ERROR: Incompatible standard for the <> placed in <>
A standard that can be applied to a set depends on the type of gates pulled into it. For more information, refer to the data sheet for standards that are supported for GPIO and DDRIO.

D.6.3 Fatal Errors

Table D-6 lists the PolarPro-specific error messages.

Table D-6: Fatal Error Messages

FATAL ERROR: Fatal Error: Drive Strength <> not supported for VCCIO<>
This error can occur if the drive strength applied on IO (GPIO/DDRDQ/DDRDQS) is not supported at VCCIO <>. Refer to the datasheet for more details.
FATAL ERROR: Pull down not available in <>
This error occurs if a design has a pull down constraint set on DDRIO for the BR device.
FATAL ERROR: Gate <> having Net <> driving port <> can drive only TA1,TA2,TB1,TB2
This error occurs when the TIN output is driving IN ports other than those given in the message.
FATAL ERROR: Gate <> having Net <> driving port <> can drive only BA1,BA2,BB1,BB2
This error occurs when the BIN output is driving IN ports other than those given in the message.
FATAL ERROR: Gate <> driving dqs_shift is non DQS
This error occurs if the dqs_shift signal is driven by a non-DQS gate <>.
FATAL ERROR: dqs_shift of <> is not driven by any dqs gate
If dqs_shift is not driven by any DQS cell.
FATAL ERROR: User Net <> connected with port <> should be either GND or VCC
This error occurs if default port is connected to a net other than VCC/GND.
FATAL ERROR: Clock net <> and CCM net <> both are driving load, remove Clock net load and try
In PolarPro, the clock pad and CCM share the same global clock network in the device. This error occurs if the design's clock pad net and CCM out are connected to load.
FATAL ERROR: Clock net <> is not driving CCM load
This error occurs if the clock pad is not driving a dedicated CCM input.
FATAL ERROR: Gate <>, Slew Rate value exceeds beyond the range
Slew rate that can be set on the gate is exceeding the maximum allowed value. For more information, refer to the I/O data sheet.

© 2008 QuickLogic Corporation

Table D-6: Fatal Error Messages (Continued)

FATAL ERROR: Gate <>, Drive Strength value exceeds beyond the range
Drive strength value set on the gate is exceeding the maximum allowed value. For more information, refer to the I/O data sheet.
FATAL ERROR: RAM. <>, true dual port configuration/vertical concatenation cannot allow x36 configuation
Check the rule in which the true dual port configuration cannot allow x36 configuation. This rule is applicable only to RAM.
FATAL ERROR: Logic Gate <> has total in-outs <> which exceeds the maximum limit <> the design is non-routable
Design is using a logic gate macro that uses maximum number of in/outs. If allowed to run through the router, the design will not be routable.
FATAL ERROR: Net <>, connecting to both routable and non-routable ports
Design has a net connected to a hardwired I/O port as well as connecting to other routable port of I/Os.
FATAL ERROR: Number of DO's cannot exceed <> DQ in a set
In PolarPro, the number of DQ gates driven by a single DQS is restricted to 11.
FATAL ERROR: DQ <> driven by <> are placed in <> with <>
The restriction that DQs driven by DQS should be pulled/fix placed in the same set is violated.
FATAL ERROR: Invalid connection from DQS to <> of type <>
This error occurs if DQS is driving any gate other than DQ.
FATAL ERROR: Gate can't exists in single instance
This error occurs if the design is trying to use the differential clock with only clock plus or clock minus gate instance.
FATAL ERROR: <> and <> mapped to two different sets <> <>
This error occurs if clock plus and clock minus gates are mapped to two different sets.
FATAL ERROR: Number of clock gates mapped to a set are more than available
In PolarPro devices, differential clock gates that a set can accommodate are restricted. This error occurs if the number of gates used is exceeded.
FATAL ERROR: Number of gates mapped <> to set exceed available limit <>
This error occurs when the number of gates pulled into a set is higher than what is available.

© 2008 QuickLogic Corporation

D.7 SpDE PolarPro Tool Verifier

D.7.1 Fix Placement

Table D-7 lists the PolarPro-specific Fix Placement messages.

Table D-7: Fix Placement Messages

WARNING: GPIO Gate mapped on DDRIO cell cannot be fix placed on GPIO cell
In the current version, a GPIO mapped to DDRIO cannot be mapped to GPIO.
WARNING: DDR-DQS IOs cannot be moved as DQ cannot be placed on Differential Clock
This message appears when trying to move a DDR set to another set where one of the DQ locations is occupied by a differential clock. This can be fixed by moving the differential clock to any other set and then moving DQS.
WARNING: DDR-DQ IOs can only be fix placed in a set where DDR-DQS is placed
First move DQS and then move DQ within the set.
WARNING: DDR-DQ IOs cannot be fix placed before fixing DDR-DQS location
First move DQS and then move DQ within the set.
WARNING: DDR-DQ IOs can only be fix placed within the set
DQ I/Os cannot be moved outside of the set.
WARNING: DDR-DQ IOs cannot be placed on differential clock set
DQ I/Os cannot swap with the differential clock and vice-versa.
WARNING: DDR Differential Clock cannot be mapped on used DQ IO
DQ I/Os cannot swap with the differential clock and vice-versa.
WARNING: Trying to fix place DDR Differential Clock Plus cell on invalid location, placement not allowed
Differential clock locations are fixed; they cannot be fix placed at arbitrary locations.
WARNING: Trying to fix place DDR Differential Clock on used DDR IO cell, not allowed
Differential clock locations are fixed; they cannot be fix placed at arbitrary locations.
WARNING: DDR Differential Clock cannot be moved as target IO is used as DDR-DQ
Move the DQS out of this set and then move the differential clock.

D.7.2 Timing Driven Placer

Table D-8 lists the PolarPro-specific Timing Driven Placer messages.

Table D-8: Timing Driven Placer Messages

WARNING: IO Bank Mapping Failed
Could not pull one or more I/Os into the assigned bank. Recreate (verify) the Bank>Gate mapping in the I/O Configuration tool.

© 2008 QuickLogic Corporation

D.7.3 RAM ROM Wizard

Table D-9 lists the PolarPro-specific RAM ROM Wizard messages.

Table D-9: RAM ROM Wizard Messages

WARNING: Dual Port not allowed for this depth and width
Dual port mode is not allowed for a given configuration.
WARNING: Dual port is allowed only for depth > 512 and width > 9
Dual port mode is allowed only for the following configurations: 512x18 and 1024x9.

D.7.4 Power Calculator

Table D-10 lists the PolarPro-specific Power Calculator messages.

Table D-10: Power Calculator Messages

EXIT: pcCntPNet: loop detected
Internal error – contact QuickLogic Support.
WARNING: Parent not found
Internal error – contact QuickLogic Support.
WARNING: Power calculator data has syntax errors.\n
Internal error – contact QuickLogic Support.
WARNING: Power calculator data cannot be loaded.\n
Internal error – contact QuickLogic Support.
WARNING: Unknown option %s in section Main has been ignored.\n, pOptionName
Please verify the input file or check for examples in the Help section *Power Calculator Input File.*
WARNING: Invalid value [%s] given for AvgFreq, in the section %s, has been ignored, pValue, pSectionName
Please verify the input file or check for examples in the Help section *Power Calculator Input File*
WARNING: Invalid value [%s] given for Count, in the section %s, has been ignored, pValue, pSectionName
Please verify the input file or check for examples in the Help section *Power Calculator Input File*
WARNING: Invalid value [%s] given for Count, in the section %s, has been ignored
Please verify the input file or check for examples in the Help section *Power Calculator Input File*
WARNING: Invalid value [%d] given for Input-Count, for the clock-%s, has been ignored
Please verify the input file or check for examples in the Help section *Power Calculator Input File*
WARNING: Invalid value [%d] given for Output-Count, for the clock-%s, has been ignored
Please verify the input file or check for examples in the Help section *Power Calculator Input File*
WARNING: Invalid values given for Input and Output Component Counts %d,%d are ignored for the clock-%s
Please verify the input file or check for examples in the Help section *Power Calculator Input File*
WARNING: Invalid value [%s] given for SystemFreq, in section %s, has been ignored
Please verify the input file or check for examples in the Help section *Power Calculator Input File*

© 2008 QuickLogic Corporation

D.7.5 Logic Optimizer

Table D-11 lists the PolarPro-specific Logic Optimizer messages.

Table D-11: Logic Optimizer Messages

EXIT: lrqDDRMportTypeGPIOMportType: unknown GPIO Mport
Internal error – contact QuickLogic Support.
EXIT: lr: translate3inLutPortType
Internal error – contact QuickLogic Support.
EXIT: lr: lrE3MoveFzToTzLut
Internal error – contact QuickLogic Support.
EXIT: lr: routeOutputToCz: unknown port type
Internal error – contact QuickLogic Support.
EXIT: lr: buildE3TempOutputs: unknown port type
Internal error – contact QuickLogic Support.
WARNING: Port %s is Not B-Frag Mport, dbqMportType(mtype)
Internal error – contact QuickLogic Support.
ERROR: lrBuildE3InvPort: %s, dbgMportName(dbqMportType(lrgCPortType(cport)))
Internal error – contact QuickLogic Support.

© 2008 QuickLogic Corporation

Appendix E
Setting Up Third-Party Simulators

This appendix provides information on setting up third-party VHDL/Verilog Simulators. It contains the following sections:

- **"VHDL Simulator Functional Overview" on page 547**
- **"Verilog Simulator Functional Overview" on page 547**
- **"ModelSim for PC Platform" on page 548**
- **"Active HDL for PC Platform" on page 549**
- **"Synopsys VSS for Unix Platform" on page 550**

E.1 VHDL Simulator Functional Overview

In order to perform timing simulation of a design using a third-party VHDL simulator, the following two requirements must be met:

- Simulator must be IEEE Vital-VHDL compliant.
- QuickLogic primitive library for timing simulation must be set up for the simulator.

Depending on the device targeted for the post layout simulation, the following primitive libraries have to be added:

- Pre-0.25 micron family devices—copy the contents of `qlvtl95-original.vhd` to `qlvtl95.vhd` before starting simulation
- 0.25 micron family devices—copy the VITAL library of `qlvtl95-25.vhd` to `qlvtl95.vhd`
- PolarPro family devices—use `qlvtlpp.vhd`
- PolarPro II family devices—use `qlvtl.vhd`

These libraries are located in the default directory `pasic\spde\data`, where QuickWorks is installed. For more information on how to set up libraries, refer to your simulator's user manual. PolarPro II libraries are located in the default directory `pasic\spde\data\PolarPro-II`.

E.2 Verilog Simulator Functional Overview

For the timing simulation of Verilog designs, the designer has to include the QuickLogic primitive library in the workspace. The library file should be added based on the targeted device family:

- Pre-0.25 micron family devices—copy the contents of `qlprim-orginal.v` to `qlprim.v` before starting simulation
- 0.25 micron family devices—copy `qlprim-25.v` to `qlprim.v`
- PolarPro family devices—use `qlprimpp.v`
- PolarPro II family devices—use `qlprim.v`

© 2008 QuickLogic Corporation

The primitive libraries for timing simulation are located in the default directory `pasic\spde\data`, where QuickWorks is installed. The primitive libraries for PolarPro II are located in the default directory `pasic\spde\data\PolarPro-II.`

This appendix will demonstrate how to setup the QuickLogic primitive library. Once it is setup, all future simulations will automatically have access to the primitive library.

The following three examples are provided to illustrate the library setup procedure:

- ModelSim on a PC platform
- ALDEC on a PC Platform
- Synopsys VSS on a UNIX platform

E.3 ModelSim for PC Platform

NOTE: Use version 4.6g or later for VITAL 3.0 compliance.

This example assumes that QuickWorks is installed on `c:\pasic`, and that ModelSim is installed on `c:\Modeltech_<version_num>.`

1. Make a qlmodel directory in `c:\Modeltech_<version_num>` by typing:

   ```
   qlmodel
   ```

2. Copy the required primitive library file depending on whether the HDL is Verilog/VHDL from the `\pasic\spde\data` directory to the `qlmodel` directory created above.
3. To invoke ModelSim, double-click on the **ModelSim SE <version_num>** icon.
4. From the ModelSim transcript dialog box, change to the `qlmodel` directory using the command line:

   ```
   cd c:\Modeltech_<version_num>\qlmodel
   ```

5. To create a new library in ModelSim, use the `vlib` command:

   ```
   vlib qlprims
   ```

6. Map this new library to its source:

   ```
   vmap qlprims c:\Modeltech_<version_num>\qlmodel\qlprims
   ```

7. To confirm the library mapping, use the `vmap` command by itself. This will show the mapping of current libraries:

   ```
   vmap
   ```

8. Finally, compile the mapped library and its source file for use with logical library work:

   ```
   vcom -work qlprims <VITAL VHDL model/Verilog model>
   ```

   NOTE: Once you have set up the primitive library, you will not have to do it again.

A design file (`.VHQ` for VHDL and `.VQ` for Verilog) is compiled in ModelSim with the following command:

- For VHDL:

  ```
  vcom design.vhq
  ```

- For Verilog:

  ```
  vcom design.vq
  ```


© 2008 QuickLogic Corporation

To specify the SDF file to use for timing simulation, use the following command in the transcript window:

```
vsim -t ps -sdfmax /u1=design.sdf config_name
```

where `vsim` has the following command options:

- **-t ps**—specifies to use pico-second as the simulation time resolution.
- **sdfmax**—specifies max (worst-case) timing simulation.
- **/u1=design.sdf**—specifies the instance under simulation, `u1` is the component label used in the testbench; `design.sdf` is the SDF file.
- **config_name**—specifies the configuration name of the configuration declaration in the testbench.

NOTE: For more information on using the `vsim` command, type `vsim -help` at the command line to get a detailed list of the options and switches.

E.4 Active HDL for PC Platform

For post-layout simulation performed in ALDEC, follow these steps:

1. Create a design directory using the command:

   ```
   createdesign <DesignName>
   ```

2. Create a work library using the command:

   ```
   alib work
   ```

3. Set the workspace using the command:

   ```
   set worklib work
   ```

4. Add the required source files for the post-layout simulation and compile them.

   For a VHDL design, the design `.VHQ` file and the QuickWorks primitive library in the VITAL model have to be added using the following commands:

   ```
   addfile –VHDL qlvtl95.vhd
   vlog qlvtl95.vhd
   addfile -VHDL design.vhq
   vlog design.vhq
   ```

   For a Verilog design, the design `.VQ` file and the primitives library needs to be added using the following commands:

   ```
   addfile –verilog qlprimpp.v
   vlog qlprimpp.v
   addfile -verilog design.vq
   vlog design.vq
   ```

5. To specify the SDF file needed for timing simulation the following command is used:

   ```
   asim –sdf<max/min/typ> m=designname.sdf -t <timing resolsution> <module name>
   ```

© 2008 QuickLogic Corporation

where SDF<MAX/MIN/TYP> annotates VITAL cells in the specified <region> with minimum, typical or maximum (sdfmin, sdftyp or sdfmax) timing values from the sdf_file.

E.5 Synopsys VSS for Unix Platform

NOTE: Use version 3.5a or later for VITAL 3.0 compliance.

This example assumes that you have experience in using UNIX, as well as the Synopsys VSS text editor. It also assumes that the .synopsys_vss.setup file is in your home directory, and a WORK directory has been setup already for VSS.

1. Create a QLMODEL directory:

   ```
   mkdir ~/QLMODEL
   ```

2. Copy the qlvtl95.vhd file from your spde directory to the QLMODEL directory.

   ```
   cp $SPDE_ROOT/data/qlvtl95.vhd QLMODEL
   ```

3. Change to the QLMODEL directory:

   ```
   cd QLMODEL
   ```

4. Create a directory called QLPRIMS:

   ```
   mkdir QLPRIMS
   ```

5. In your home directory edit the .synopsys_vss.setup file with a text editor such as *vi* or *emacs*.
6. Add the following four lines to your .synopsys_vss.setup file. If the last two lines already exist, just add in the first two lines:

   ```
   QLPRIMS > QUICK
   QUICK : ~/QLMODEL/QLPRIMS
   SDFNAMINGSTYLE=VITAL0
   SDFWILDCARD=TRUE
   ```

7. After editing your .synopsys_vss.setup file, change the directory to QLMODEL:

   ```
   cd QLMODEL
   ```

8. Compile the library for use with your logical library WORK:

   ```
   vhdlan -w QUICK qlvtl95.vhd
   ```


© 2008 QuickLogic Corporation

Appendix F
QuickWorks Icons

This appendix provides a complete set of QuickWorks icons. It contains the following sections:

- **"QuickWorks Windows Icon Group" on page 551**
- **"SpDE/Toolbar Menu Icon Group" on page 551**

F.1 QuickWorks Windows Icon Group

When QuickWorks is installed, a group of icons described in **Table F-1** is created.

Table F-1: QuickWorks Icons

Icon	Description
SpDE	**SpDE (Seamless pASIC Design Environment)** is the framework in which all design flows are managed. From within SpDE, all QuickWorks tools can be launched, designs can be evaluated, and devices can be programmed. SpDE supports all families of devices, in all available packages.
SCS Executive	**SCS Executive** can be launched from within SpDE to enter the schematic design flow. QuickLogic does not recommend running SCS as a stand alone application. The SCS Executive allows access to all SCS design entry tools: Schematic Editor, Symbol Editor, Hierarchy Navigator, and Waveform Editor.
Precision RTL	**Precision RTL Synthesis** by Mentor graphics is a synthesis platform the maximizes the performance of both existing programmable logic devices (CPLDs and FPGAs) and next-generation, multi-million gate field programmable system-on-chip (FPSoC) devices.

F.2 SpDE/Toolbar Menu Icon Group

Table 7: SpDE Menu Icons

Icon	Equivalent Menu Command	Function
	File > Open	Imports .QDS, .QDF, or .CHP files
	File > Save	Saves the design to a .CHP file
	File > Print	Prints the current design Physical View

© 2008 QuickLogic Corporation

Table 7: SpDE Menu Icons (Continued)

Icon	Equivalent Menu Command	Function
	View > Zoom In	Zooms in on the Physical View
	View > Zoom Out	Zooms out on the Physical View
	View > Pan	Pans around the Physical View
	View > Full Fit	Displays entire chip in SpDE dialog box
	Design > Text Editor	Turbo Writer/HDL Editor
	Design > Schematic Editor & Navigator	SCS Design Entry
	Tools > Options	Setup Options for SpDE tools
	Tools > Run Selected Tools...	Runs a selection of tools, such as Place, Route etc.
		Cancels Progress
	Tools > Static Timing Analyzer	Analyzes path delay, sets constraints and displays timing information
	Tools > Constraints Manager	Opens the Constraints Manager dialog box
	Tools > Timing Analysis	Opens the Timing Analysis tool for PolarPro devices
	Tools > RAM/ROM/FIFO Wizard	Opens the RAM Module Creation Wizard
	Design > Active HDL Simulator	Launches Active HDL Simulator
	Tools > Power Calculator	Opens the Power Calculator dialog box
	Info > Reporting... > Report File	Opens the design report file (.rpt) in HDL Turbo Writer

© 2008 QuickLogic Corporation

Table 7: SpDE Menu Icons (Continued)

Icon	Equivalent Menu Command	Function
	Help > About SpDE	Opens the About SpDE dialog box
	Help > SpDE	Launches SpDE Help
	Help > Web Update	Lists all of the updates available for the installed version from the web
		Starts the design flow interface
	Tools > Migrate Part	Starts the dialog for migrating a design to another device

© 2008 QuickLogic Corporation

Appendix G
Synthesis Tool Support: Leonardo Spectrum

This appendix describes Mentor Graphics' Leonardo Spectrum™ tool and contains the following sections:

- **"Functional Overview" on page 555**
- **"Using Leonardo Spectrum" on page 556**
- **"Tcl Script Support" on page 558**
- **"EDIF Design in QuickWorks" on page 559**
- **"Writing HDLs for Leonardo Spectrum Synthesis Tool" on page 559**
- **"Specific Settings to be Selected in Leonardo Spectrum" on page 559**

G.1 Functional Overview

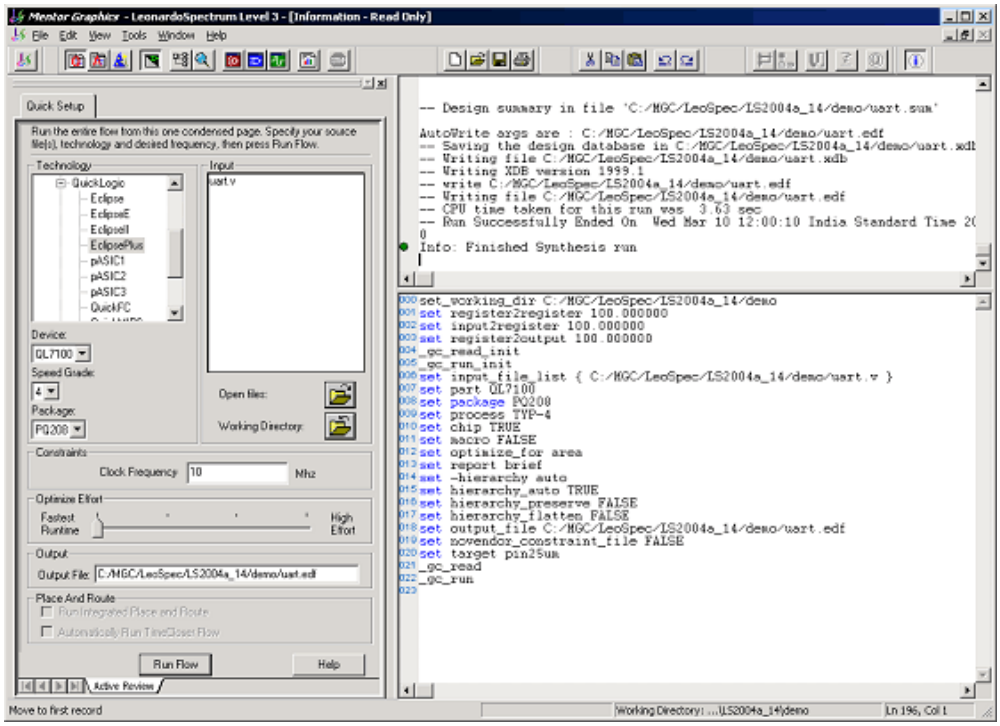
The QuickWorks tool supports QDIF and EDIF input file formats. EDIF is a general file format, used by multiple synthesis tools. The Leonardo Spectrum tool supports QuickWorks macro library. And the EDIF output from Leonardo Spectrum can be loaded in QuickWorks.

© 2008 QuickLogic Corporation

G.2 Using Leonardo Spectrum

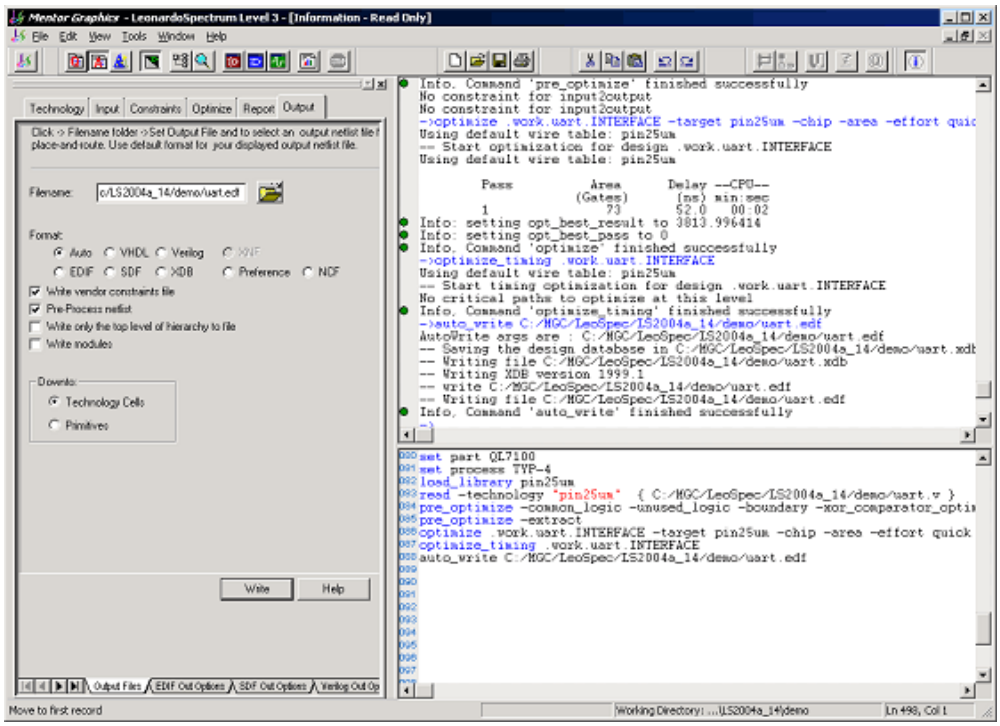
The Leonardo Spectrum tool can be used in Quick Setup (**Figure G-1**) or Flow Tabs (**Figure G-2**) mode (from the Tools menu). The Flow Tabs option is useful when multiple options are to be set.

Figure G-1: Leonardo Spectrum in Quick Setup Mode



© 2008 QuickLogic Corporation

Figure G-2: Leonardo Spectrum in Flow Tabs Mode



The major steps involved in the synthesis process using Leonardo Spectrum are:

1. **Technology tab**: Since the Leonardo Spectrum tool supports different technology, select the QuickWorks device family and specific device, and then click Load Library.
2. **Input tab**: Add the input HDL files for synthesis. There are multiple options that can be set as needed.
3. **Constraints tab**: Specify the timing constraints for the design.
4. **Optimize tab**: There are multiple configurable options the can be used for the design optimization. Based on the design requirement, set the options before optimizing the design.
5. **Report tab**: Select the output reports.
6. **Output tab**: The Output generation step also has multiple options. Set the output based on the requirements, and click **Write** to get the output.

The Leonardo Spectrum output includes an output file, report file (if specified), and log file (`Exemplar.log`).

© 2008 QuickLogic Corporation

G.3 Tcl Script Support

The Leonardo Spectrum tool has Tcl script support. The designer can create a tcl script and run it using the Leonardo Spectrum tool **File>Run Script** option to synthesis a design.

The tool also has command line support, and the Tcl script can be run as:

`Spectrum.exe -file <tcl_script>`

```
set part QL5032-33
set package PQ208
set process WC-A
set chip TRUE
set macro FALSE
load_library qram
set_working_dir C:/pasic/exemplar/reference/pci5032_208/verilog
set optimize_for area
set report brief
set -hierarchy auto
set resource_sharing FALSE
set maxdelay 0
set effort quick
set hierarchy_auto TRUE
set hierarchy_preserve FALSE
set hierarchy_flatten FALSE
read -technology "qram" -dont_elaborate  { pci5032_208.v }
elaborate pci5032_208
set_attribute .work.pci32.INTERFACE -name NOOPT -value "1"
set_attribute .work.pci32.INTERFACE -name DONT_TOUCH -value "1"
set process WC-
set process WC-A
set edifout_power_ground_style_is_net TRUE
optimize .work.pci5032_208.INTERFACE -target qram -chip -area -effort quick -
hierarchy auto
report_area pci5032_208_edif.rpt -cell_usage -all_leafs
set output_file pci5032_208.edf
set novendor_constraint_file FALSE
auto_write pci5032_208.edf
save_project_script pci5032_208.scr
```


© 2008 QuickLogic Corporation

G.4 EDIF Design in QuickWorks

The EDIF output generated by using Leonardo Spectrum can be loaded in QuickWorks by selecting the **File>Open** option. Multiple tool options can be run on the loaded design and a placed, routed chip file can be saved.

G.5 Writing HDLs for Leonardo Spectrum Synthesis Tool

A pure behavioral HDL can be written to be synthesized using Leonardo Spectrum. The designer must make sure that proper constraints, identified by Leonardo Spectrum, are used in the HDL, based on requirement.

When the HDL is generated using the QuickWorks tool Hierarchy Navigator, the HDLs has all the constraints set for the Leonardo Spectrum tool. So the designer need not worry about the constraint for such HDLs.

G.6 Specific Settings to be Selected in Leonardo Spectrum

Since QuickWorks does not support EDIF arrays, while generating output from Leonardo Spectrum, make sure the **Allow Writing Buses** option is not selected.

Select the **Write out power and ground as undriven nets with special names** option. With this option, Leonardo Spectrum dumps the nets VCC and GND, wherever needed. They are identified by QuickWorks.

NOTE: By default, for QuickLogic devices:
– The following options are set false, as they don't support arrays in EDIF netlist:
`edif_write_arrays, vhdl_write_arrays, verilog_write_arrays`
– The `edifout_power_ground_style_is_net` option is enabled.

© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation

Appendix H
Precision RTL Synthesis Tool

This appendix describes the *Precision RTL Synthesis* tool from Mentor Graphics that provides support for QuickLogic PolarPro, PlolarPro II, Eclipse, Eclipse II, Eclipse Plus, and QuickPCI devices.

- **"Functional Overview" on page 561**
- **"Using Precision RTL Synthesis" on page 561**
- **"Tcl Script Support" on page 564**
- **"Loading EDIF in QuickWorks" on page 564**

H.1 Functional Overview

Precision RTL Synthesis supports the EDIF format and generates the following files:

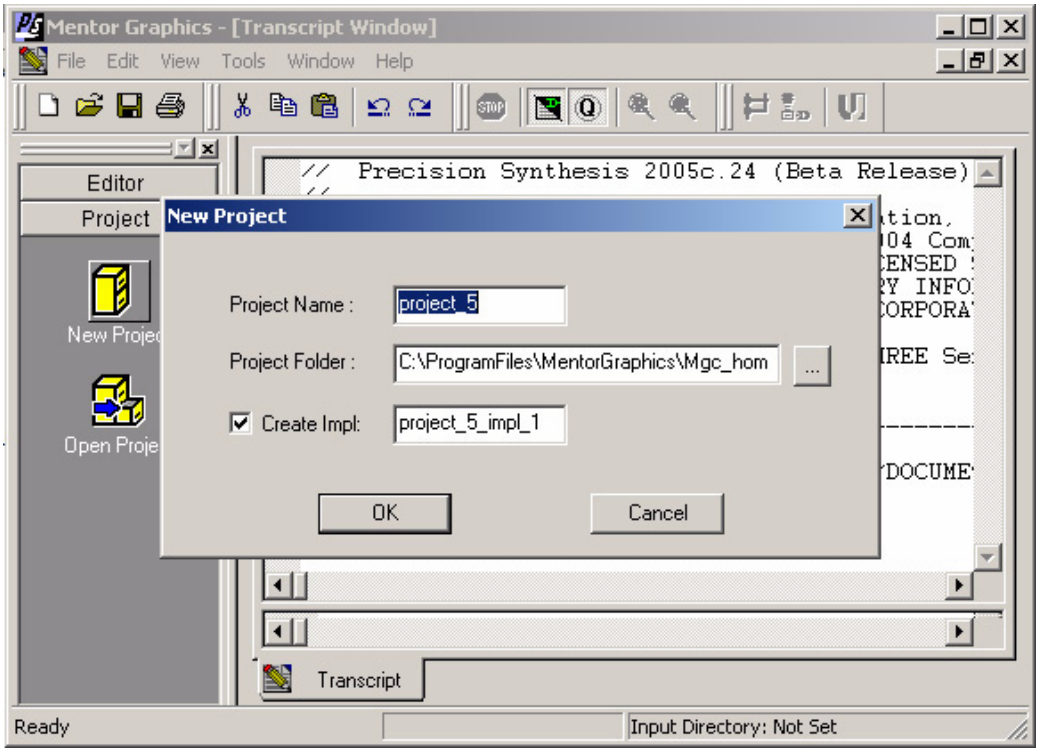
- **LUT based netlist** When the HDL passed to the synthesis tool is in the form of behavioral level.
- **Macro based netlist** When the HDL passed to the synthesis tool is generated from the Hierarchy Navigator of QuickWorks Tool or the HDL is written as a netlist of the macros from the QuickLogic Library.

H.2 Using Precision RTL Synthesis

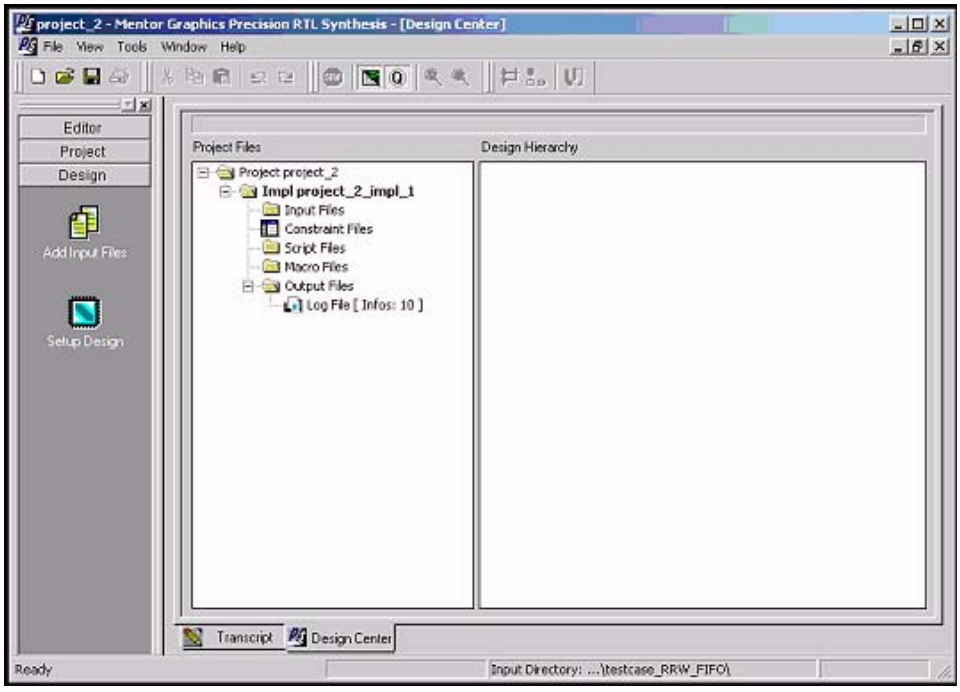
To start using the Precision RTL Synthesis tool, follow the steps below:

© 2008 QuickLogic Corporation

1. Open Precision RTL Synthesis. The New Project dialog box appears.



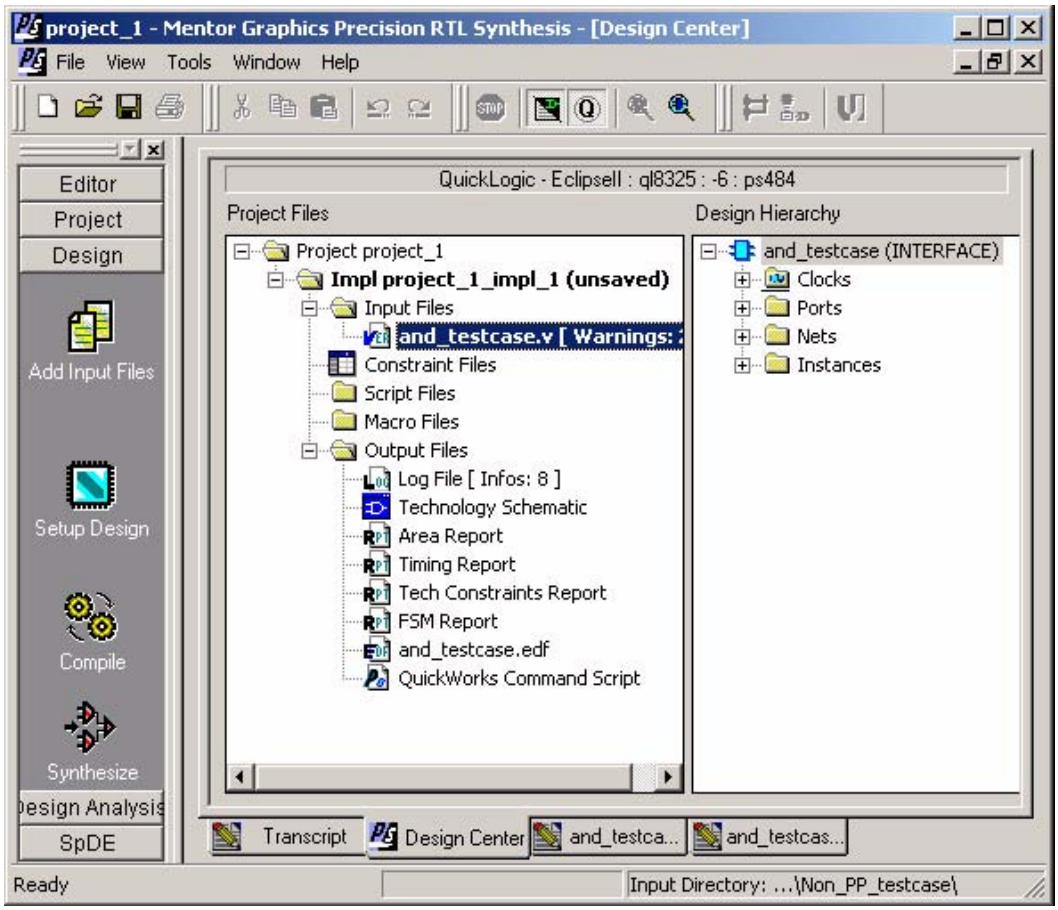
2. Type in the name of your project and specify the location where the implementation is to be saved (or browse for the location).
3. Select **Create Impl** and provide the name of the implementation folder.



4. To select the target device, click **Setup Design** in the design bar.

© 2008 QuickLogic Corporation

5. To add the required source files, click **Add Input Files** in the design bar.
6. Once the input file and the required constraint are added, click **Synthesize** in the design bar to generate the EDIF. The Project Navigator provides access to all of the input files, the output file, the RTL schematic, and the technology schematic.



Precision RTL Synthesis auto-generates a TCL, which can be used to run the SpDE from command line or using the graphical interface. The script loads the generated EDIF netlist and runs all SpDE tools.

© 2008 QuickLogic Corporation

H.3 Tcl Script Support

The designer can write a TCL script and run the TCL script through graphical interface by selecting **File>Run Script**, or through the command line using the following command:

```
Precision.exe –shell –file <filename.tcl>
```

The following is a sample list of TCL commands used to run the Precision RTL Synthesis tool:

```
setup_design -manufacturer QuickLogic -family PolarPro -part ql1p100 -package
pf144 –speed 7
add_input_file { testcase_add4.vhd  C:/pasic/spde/data/macros_pp.vhd }
setup_design -design testcase_add4 -architecture SCHEMATIC
compile
synthesize
auto_write testcase_add4.edf
save_impl
```

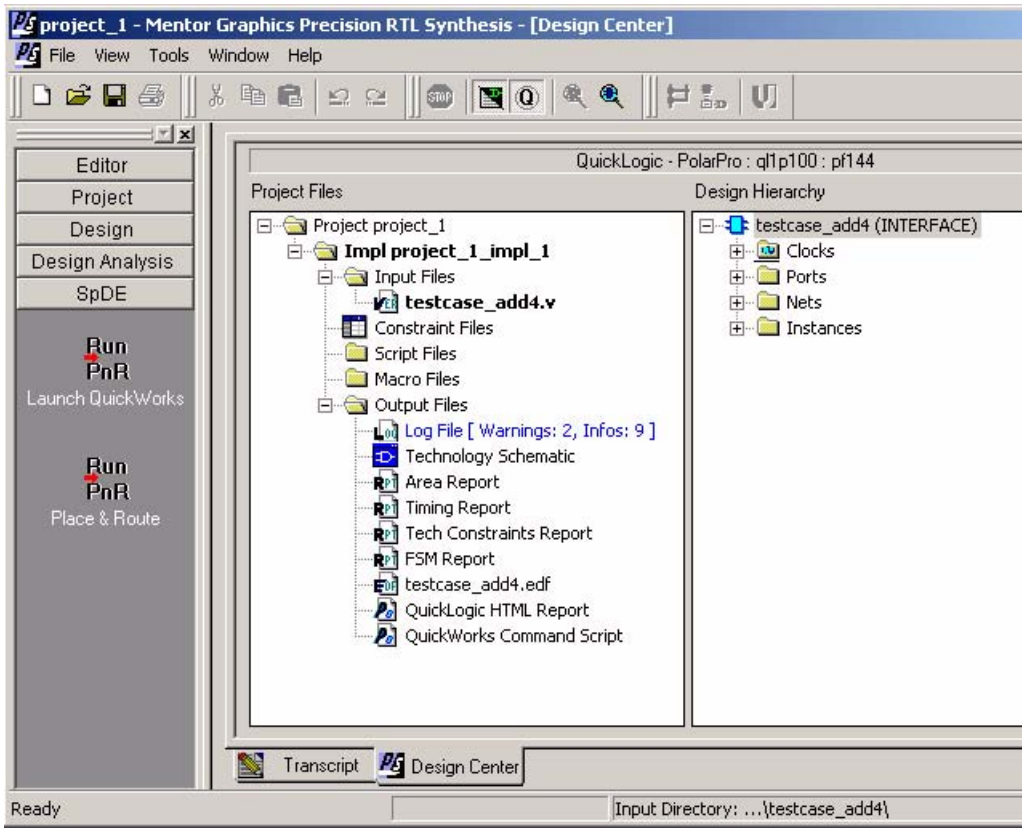
H.4 Loading EDIF in QuickWorks

QuickWorks is integrated with the Precision RTL Synthesis tool. To load the generated EDIF in QuickWorks:

1. From Precision RTL Synthesis, select **File>Open**.
2. Set the option in the Precision RTL Synthesis to invoke the SpDE by selecting **Tools>SetOptions>SpDE>Launch QuickWorks>Set the path of the QuickWorks Installation**

© 2008 QuickLogic Corporation

Precision RTL Synthesis generates a TCL script along with the EDIF netlist to run the QuickWorks tool (`*_qlogic.tcl`).



3. Select **Launch QuickWorks** in the SpDE bar to invoke QuickWorks and run the TCL script generated by Precision RTL Synthesis.
4. From QuickWorks, select **Tools>Run TCL Script**.

© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation

Appendix I
QuickTools: UNIX Version of QuickWorks

QuickTools is the UNIX version of QuickWorks. This appendix covers everything you need to know about QuickTools and contains the following sections:

- **“Installation Instructions” on page 567**
- **“Installing QuickTools Server” on page 569**
- **“Installing QuickTools Client” on page 572**
- **“System Requirements - Solaris” on page 573**
- **“System Requirements - Linux” on page 573**
- **“Operating System Patches” on page 575**
- **“Utilities on UNIX Workstation” on page 577**

NOTE: Device Programming is not available on UNIX workstations. A completed `.CHP` (chip) file must be transferred to a PC running the QuickLogic Programmer Kit software in order to program devices.

I.1 Installation Instructions

QuickTools is available for Sun/Solaris and Linux versions of UNIX. The following sections describe the installation procedure for each of those platforms.

I.1.1 Installation for Sun/Solaris

1. Mount the QuickTools CD-ROM.
   If your system supports automatic mounting, skip this step. (Mounting the CD usually requires superuser privileges. Contact your local system administrator for assistance.)

   ```
   > mount -F hsfs -o ro /dev/dsk/(device_name) /(directory)
   ```

2. Execute the installation script:

   ```
   > cd (mounted_cdrom_directory)/
   > ./install.cd
   ```

3. Update your `.login/.profile` and `.cshrc/.bashrc` files according to the directions indicated at the end of installation.

I.1.2 Installation for Red Hat Linux

1. Mount the QuickTools CD-ROM.
   If your system supports automatic mounting, skip this step. (Mounting the CD usually requires superuser privileges. Contact your local system administrator for assistance.)

   ```
   > mount /dev/cdrom /(directory)
   ```

2. Execute the installation script:

   ```
   > cd (mounted_cdrom_directory)/
   > ./install.cd
   ```


© 2008 QuickLogic Corporation

QuickTools installation for Linux workstations supports the following two methods:

- **Server/Client**—The common read-only SpDE files are installed on the application server and are accessible to all users. The user-specific files are installed by the client setup in individual user directories.

NOTE: You must login as `root` to perform server installation. To setup clients, please follow the steps at the end of the server installation.

- **Standalone**—All SpDE files are installed in the user-specified location.

While running QuickTools installation, you can choose either the Server/Client installation or the Standalone installation.

3. Update your `.login/.profile` and `.cshrc/.bashrc` files according to the directions indicated at the end of installation.

© 2008 QuickLogic Corporation

I.2 Installing QuickTools Server

The Server installation starts with the following welcome message:

```
> Welcome to QuickLogic's QuickTools installation
> Version: 9.6
```

NOTE: If you are not logged in as `root`, go directly to step 2.

1. Choose between a Server or a Standalone installation. To select Server installation type **y**, otherwise type **n** or press **Enter** to continue with Standalone installation.

   ```
   > Do you want to perform server installation? [y/n], (default = n):
   ```

2. Choose whether to install the software. To install type **y** or press **Enter**, otherwise type **n** to abort the installation.

   ```
   > Do you want to install the SpDE files? [y/n], (default = y):
   ```

3. Specify your name and your company's name.

   ```
   > Getting the licensing information...
   > Enter your name:
   > Enter your company's name:
   ```

4. Specify the full path of target directory where you want to install QuickTools.

   ```
   > Where do you want to install it? [.../quicktools_95]:
   > Checking system requirements...
   ```

   Installation script performs few initial checks (i.e., disk space available on the target location to install the bare minimum components of QuickTools.

5. Select between Custom or Full installation. Type **y** to perform full installation, otherwise type **n** or press **Enter** for custom installation.
   Full installation installs all the QuickLogic devices and the related files to the target location. During Custom installation, you can choose QuickLogic devices (based on architecture) and the related files to be installed.

   ```
   > Do you want to perform full installation without architectural
   selection [y/n],(default = n):
   ```

   NOTE: If you selected Full installation, skip the following step.

   ```
   Custom installation: Select the devices to install.
   > Select the Architectures to install
   >          1. PASIC1    [*]
   >                 -       a. p8x12b [*]
   >                 -       b. p12x16b [ ]
   >                 -       c. p16x24b [ ]
   >                 -       d. p24x32b [ ]
   >          2. PASIC2      [ ]
   >          3. PASIC3      [ ]
   >          4. QuickPCI    [ ]
   >          5. QuickRAM    [ ]
   >          6. EclipsePlus [ ]
   >          7. Eclipse     [ ]
   >          8. QuickFC     [ ]
   >          9. QuickSD     [ ]
   >         10. QuickMIPS   [ ]
   ```

© 2008 QuickLogic Corporation

```
>         11. Eclipse-II  [ ]
>         12. Eclipse-E   [ ]
>
>          L. Select All
>          S. Show Summary
>          N. Install the selected devices
> Enter the selection :
```

6. By default, no devices would be selected. To install the required devices under specific families, for example type **1** to show/hide the PASIC1 devices available, and then type **a** to select/deselect p8x12b device from PASIC1 family. To select all the devices under the selected family, use the *Select All* option by typing **L**. For a summary of the devices selected, type **S**. To end the selection procedure and continue installing the selected devices, type **N**.
7. Wait while the installation script validates the system requirements and installs the selected components. You will see the following message:

```
> Preparing database for uncompressing SpDE archive...
> Validating the system requirements...
> Available Disk space ... Kbytes
> Required Disk space ... Kbytes
> Free Disk space ... Kbytes
```

The installation prepares itself to install the software, performs a few checks regarding the actual disk space required/available on the target location and continues the installation.

```
> Uncompressing SpDE archive...
> ...
```

NOTE: If you are performing a Standalone installation, skip the following step.

8. After a successful Server installation, the following message is displayed:

```
> QuickTools Server Installation completed successfully.
> Other steps required to run spde (see install instructions):
> 1) Perform client installation as follows:
>    -> Run "setup_client.csh" to setup the clients.
> NOTE: QuickTools Client Installation can be done by "root" or
> individual user.
```

The Server installation is completed.

NOTE: For details on QuickTools Client Installation, refer to **"Installing QuickTools Client" on page 572**.

9. The Standalone installation installs windows configuration file(s) to user's directory. If you are root, specify the user who will use SpDE.

```
> NOTE: Copying the .ini file to a user's directory
> Enter the user name that will use SpDE: < ... >
```

10. After successful Standalone installation, the following message is displayed:

```
> Installation completed successfully.
> Please consult the QuickTools Users Guide for instructions.
> Other steps required to run spde (see install instructions):
> 1) Setup your environment as follows:
-> If you are using C shell or one of its derivatives, add following
command in ~/.cshrc or ~/.login source .../quicktools_95/spde_env.csh
```


© 2008 QuickLogic Corporation

```
else
-> If you are using Bourne shell or one of its derivatives, add following
command in ~/.bashrc or ~/.profile .../quicktools_95/spde_env
> 2) Restart the window manager before running SpDE
```

Set up the environment variables as mentioned at the end of the installation and restart the Windows Manager. This completes the Standalone installation.

© 2008 QuickLogic Corporation

I.3 Installing QuickTools Client

The Client installation starts with the following welcome message:

```
> Welcome to QuickLogic's QuickTools client installation for $OS
> Version: 9.5
```

1. Confirm if you want to install the client. To setup the Client type **y**, otherwise type **n** or press **Enter** to abort setup.

   ```
   > Do you want to perform client installation? [y/n], (default = n):
   ```

2. Specify the full path of the Server installation directory.

   ```
   > Specify the path of the QuickTools server installation
   [.../quicktools_95]:
   ```

3. The setup will copy Windows configuration file(s) to user's directory. If you are `root`, specify the user who will use SpDE.

   ```
   > NOTE: Copying the .ini file to a user's directory
   > Enter the user name that will use SpDE: < ... >
   ```

4. After a successful client installation, the following message is displayed:

   ```
   > Installation completed successfully.
   ```

   ```
   > Please consult the QuickTools Users Guide for instructions.
   > Other steps required to run spde (see install instructions):
   > 1) Setup your environment as follows:
   -> If you are using C shell or one of its derivatives, add following
   command in ~/.cshrc or ~/.login source .../quicktools_95/spde_env.csh
   else
   -> If you are using Bourne shell or one of its derivatives, add following
   command in ~/.bashrc or ~/.profile .../quicktools_95/spde_env
   > 2) Restart the window manager before running SpDE
   ```

   Set up the environment variables as mentioned at the end of installation and restart the Windows Manager. This completes the Client installation.

© 2008 QuickLogic Corporation

I.4 System Requirements - Solaris

I.4.1 Hardware Requirements

- Sun SPARC Station 1 or higher (Sun Ultra-1 or higher recommended)
- 256 MB of RAM (512 MB recommended)
- Sun-supported mouse, monitor, and CD-ROM drive
- 1 GB of hard disk space:
  - 950 MB for QuickTools
  - 50 MB for design files

I.4.2 Software Requirements

- Solaris 2.6, 7, or 8 on the SPARC platform
- OpenGL version 1.2 (optional)
- Solaris kernel specifications as listed in **Table I-1**.

Table I-1: Kernel Specifications - Solaris

OS Version	Kernel Revision
8	January 2000 and later
7	October 1998 and later
2.6	January 1998 and late

I.5 System Requirements - Linux

I.5.1 Hardware Requirements

- Intel x86 platform
- 256 MB of RAM (512 MB recommended)
- Linux supported mouse, monitor, and CD-ROM drive
- 1 GB of hard disk space:
  - 950 MB for QuickTools
  - 50 MB for design files

I.5.2 Software Requirements

- Red Hat Linux 6.2, 7.1 on Intel x86 platform
- Korn shell: `pdksh`
- inetd: `xinetd`
- rsh server: `rsh`
- Telnet server: `telnet-server`
- FTP server: `wu-ftpd` (only required to browse UNIX file system from Windows PC)

© 2008 QuickLogic Corporation

- OpenGL (optional): Mesa, version 3.2.2
- Java SDK (optional): version 1.3
- Red Hat kernel specifications as listed in **Table I-2**.

Table I-2: Kernel Specifications - Red Hat

OS Version	Kernel Revision
6.2	2.2.14 - 5*
7.1	2.4.2 - *

where the asterisk (*) may be zero or more characters.

NOTE: While later versions of the Linux kernel should also work, they have not been fully tested.

© 2008 QuickLogic Corporation

I.6 Operating System Patches

The list of patches below is recommended for new QuickTools customers and customers who are upgrading from earlier releases of QuickTools.

CAUTION: If you are currently using QuickTools, and the installed patches for your operating system meet your needs, we recommend that you do **NOT** change your installed patches. QuickLogic makes every effort to provide the most current and accurate information about patches for the various platforms and operating systems. However, QuickLogic cannot take responsibility for patches that have not been tested.

It is recommended that you install the recommended patches before installing any other packages or patches.

Table I-3 lists compiler/runtime patches for Solaris 8 systems:

Table I-3: Patches - Solaris 8

Patch #	Reason
108434-01	Shared library fix for C++ run-time

Table I-4 list of compiler/runtime patches for Solaris 7 systems:

Table I-4: Patches - Solaris 7

Patch #	Reason
108376-16	Prevents applications from hanging on multiprocessor machines
106327-08	Shared library fix for C++ run-time
Runtime patches for specific configurations	
106144-21	Patch for Elite3D graphics card
106145-17	Patch for Creator graphics card
106146-16	Patch for M64 graphics card
106147-06	Supplemental patch for Elite3D and Creator graphics card
106148-12	Required supplemental patch for
107851-11	Patch for PGX graphics card

Table I-5 list of compiler/runtime patches for Solaris 2.6 systems:

Table I-5: Patches - Solaris 2.6

Patch #	Reason
105181-23	Fixes kernel thread problems
105633-49	Prevents applications from hanging on multiprocessor machines
106040-15	Fixes input/output problems (primarily for Japanese support)
106125-10	Improves processing of OS patches
105591-09	Shared library fix for C++ run-time
Runtime patches for specific configurations	
105360-34	Patch for Creator graphics card
105361-11	Supplemental patch for Creator/Elite3D graphics card

© 2008 QuickLogic Corporation

Table I-5: Patches - Solaris 2.6

Patch #	Reason
105362-29	Patch for M64 graphics card
105363-32	Patch for Elite3D graphics card
105492-02	Patch for cgsix graphics card
106391-01	Patch for TCX graphics card
107851-11	Patch for PGX32 (Raptor GFX)

© 2008 QuickLogic Corporation

I.7 Utilities on UNIX Workstation

Table I-6 lists some UNIX versions of common Windows utilities.

Table I-6: UNIX Utilities

Utility	Description
`mfcie`	Windows Internet Explorer ported to UNIX. `mfcie` is the Visual MainWin ported customized UNIX version of the Windows utility *iexplore*, a standard Windows utility described in Microsoft documentation. Use `mwcontrol` to set up internet options for `mfcie`.
`mwcontrol`	Graphical user interface used to set up Visual MainWin runtime system settings. `mwcontrol` is the utility behind the Visual MainWin Control Panel. Similar to the Windows Control Panel, the Visual MainWin Control Panel contains applets used to configure Visual MainWin and the computer on which Visual MainWin is running.
`notepad`	Windows Notepad ported to UNIX. `notepad` is the Visual MainWin ported UNIX version of the Windows utility *Notepad*, a standard Windows utility described in Microsoft documentation.
`winhelp`	Wrapper that calls `winhlp32`. `winhelp` is the wrapper that invokes `winhlp32`. It enables UNIX users to access Windows help files within a UNIX environment.
`winhlp32`	Windows Help utility ported to UNIX. `winhlp32` is the Visual MainWin ported UNIX version of the Windows utility *`winhlp32`*, a standard Windows utility described in Microsoft documentation. The ported version of `winhlp32` enables UNIX users to access Windows help files within a UNIX environment. `winhlp32` uses the environment variable MWINHELP_PATH to locate the full path to the WinHelp files, and the environment variable MWWINHELP_DIR to locate the full path to the directory that the WinHelp File Open dialog box opens on start-up.

© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation

Appendix J
Using ViewDraw with QuickWorks

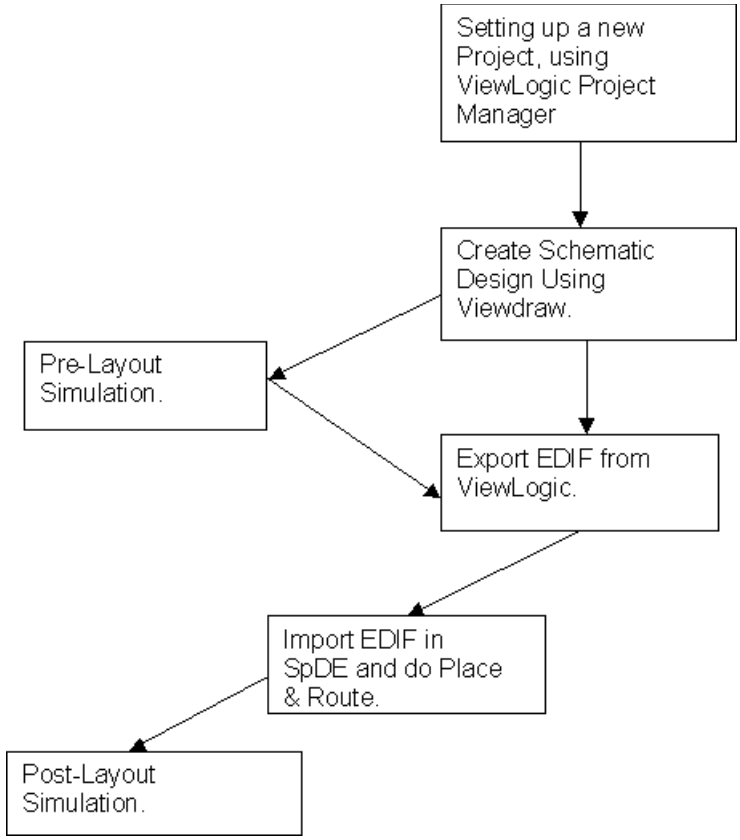
This section provides a step-by-step instruction in the design flow for ViewLogic's ViewDraw® to target QuickLogic Devices and contains the following sections:

- **"Functional Overview" on page 579**
- **"Creating New Project Using the Project Manager" on page 580**
- **"Creating a Schematic Design in ViewDraw" on page 582**
- **"Exporting in EDIF" on page 583**
- **"Import EDIF Netlist into SpDE" on page 584**

J.1 Functional Overview

Figure J-1 shows a high-level description of the design flow.

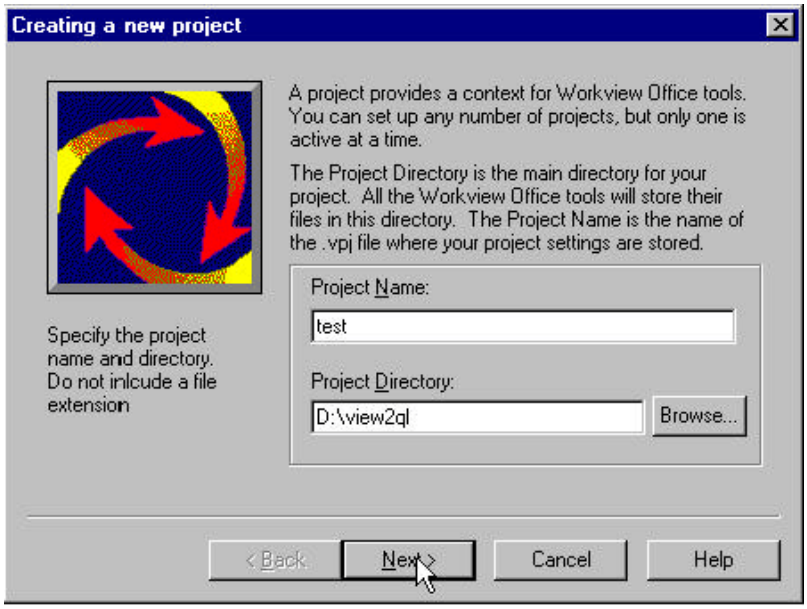
Figure J-1: QuickLogic-Viewlogic Design Flow


Setting up a new Project, using ViewLogic Project Manager
Create Schematic Design Using Viewdraw.
Pre-Layout Simulation.
Export EDIF from ViewLogic.
Import EDIF in SpDE and do Place & Route.
Post-Layout Simulation.

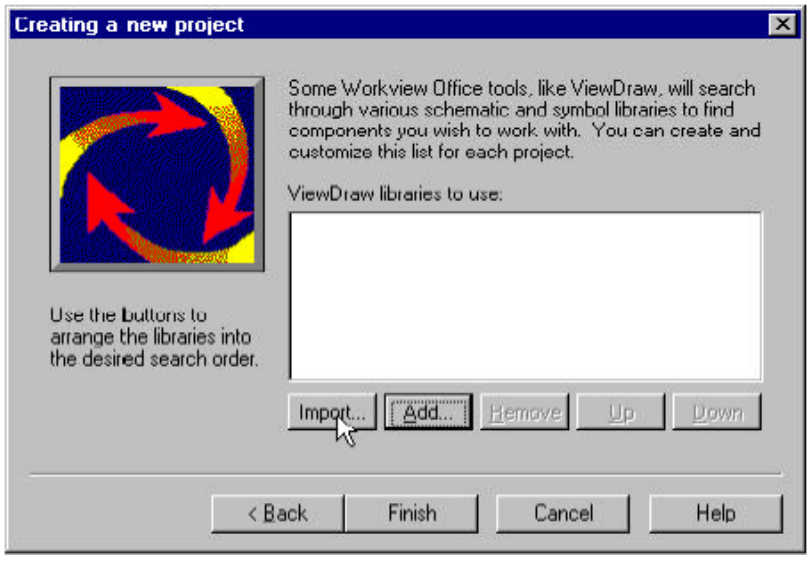
© 2008 QuickLogic Corporation

J.2 Creating New Project Using the Project Manager

1. In the ViewLogic Project Manager, click the **New Project** icon.
2. In the Project Name field, type a name for the project. In the Project Directory, type a name for the project folder or click **Browse** to select a project directory. Click **Next**.


Creating a new project
A project provides a context for Workview Office tools. You can set up any number of projects, but only one is active at a time.
The Project Directory is the main directory for your project. All the Workview Office tools will store their files in this directory. The Project Name is the name of the .vpj file where your project settings are stored.
Specify the project name and directory. Do not inlcude a file extension
Project Name:
test
Project Directory:
D:\view2ql
Browse...
< Back
Next >
Cancel
Help

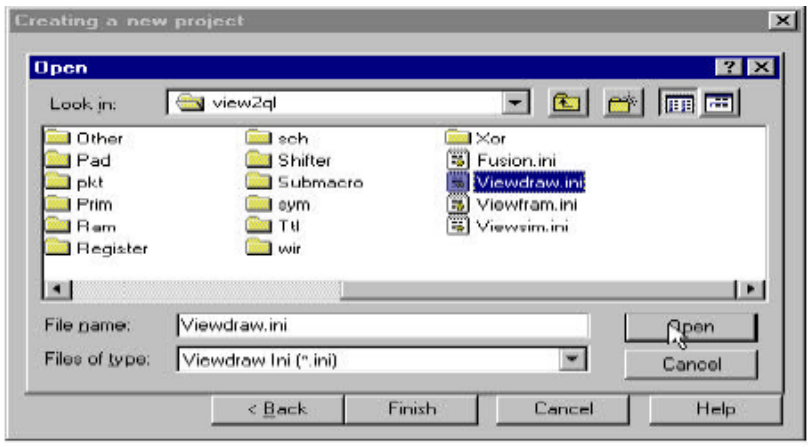
3. Click **Import**.


Creating a new project
Some Workview Office tools, like ViewDraw, will search through various schematic and symbol libraries to find components you wish to work with. You can create and customize this list for each project.
ViewDraw libraries to use:
Use the buttons to arrange the libraries into the desired search order.
Import...
Add...
Remove
Up
Down
< Back
Finish
Cancel
Help

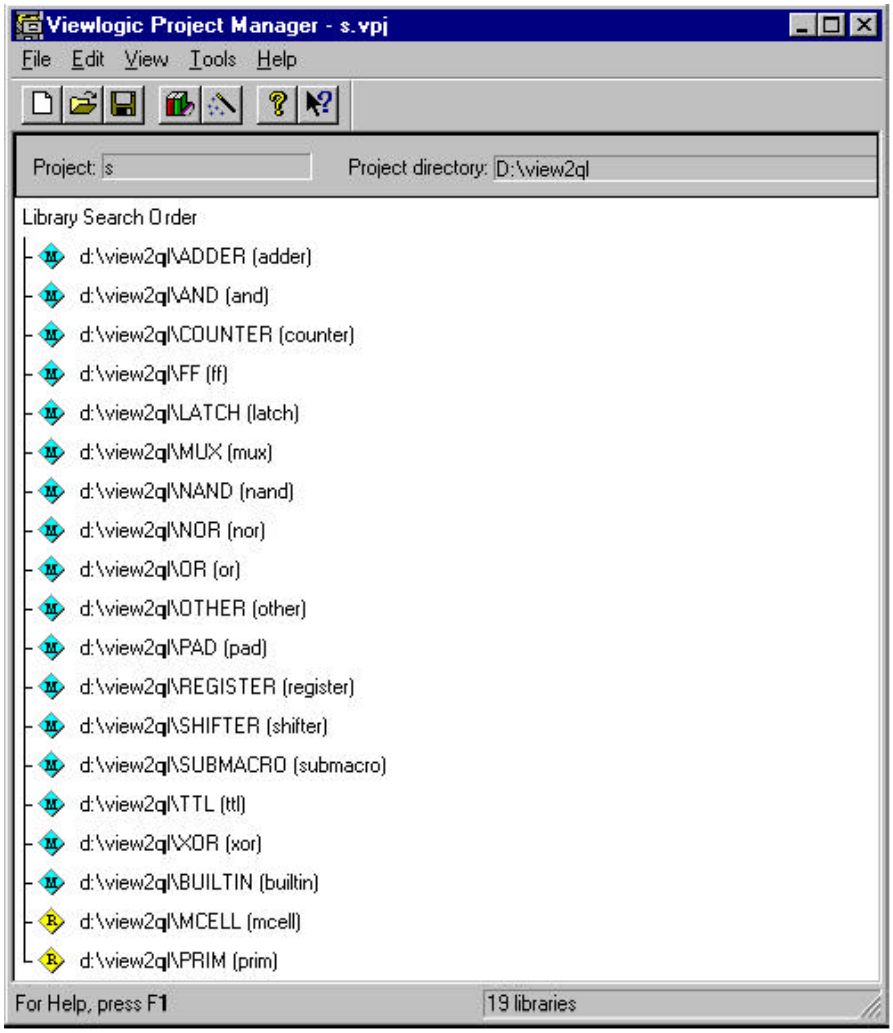
4. Select the `viewdraw.ini` library. This file should be in the folder where the `VL2QL.exe` was installed. (By default, the directory is `C:\quicklogic\`).

© 2008 QuickLogic Corporation

5. Click **Open**.



6. Click **Finish**.

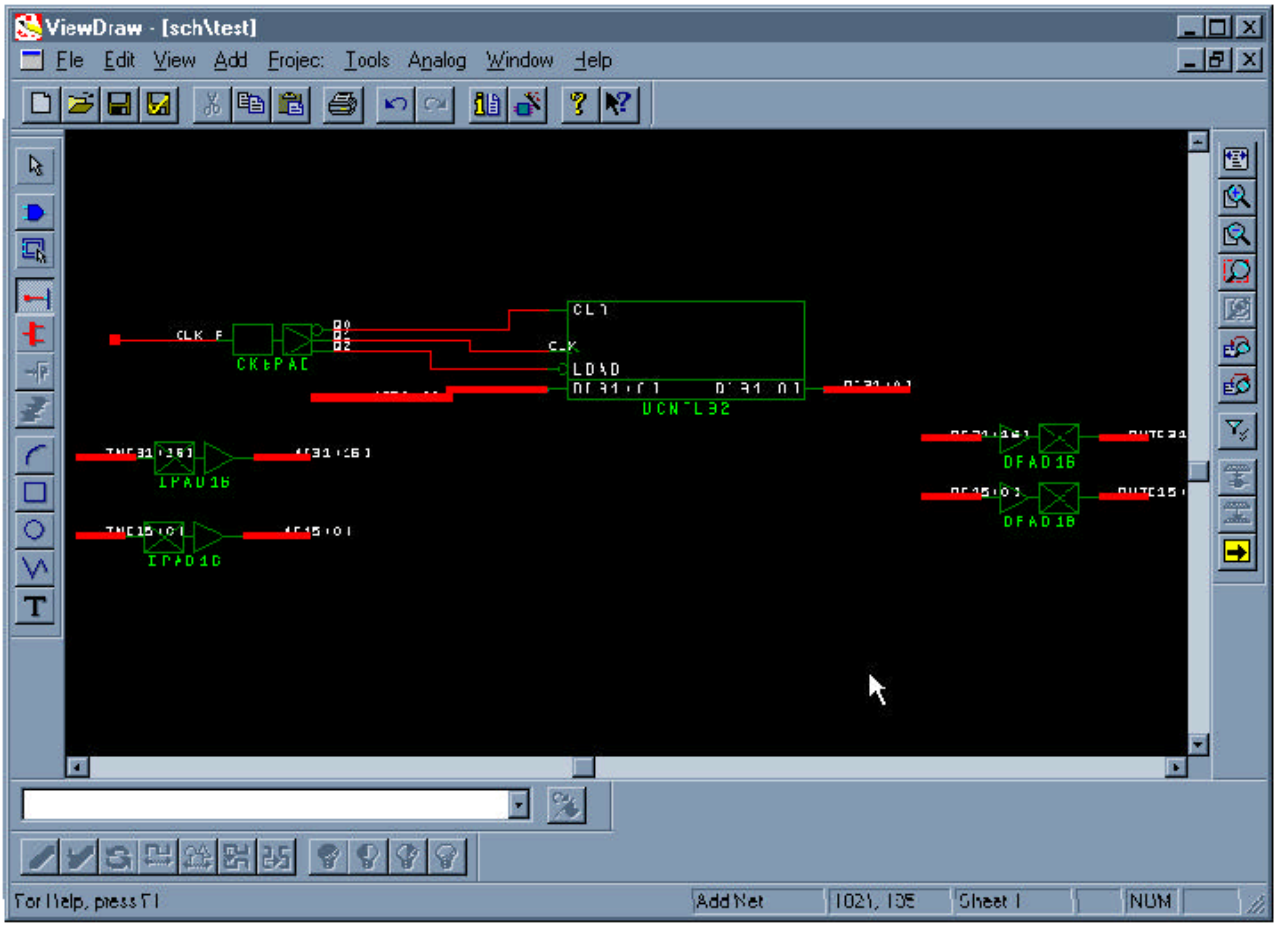


Now that you have finished setting up the project manager to use QuickLogic-specific symbols, you are ready to start your design in ViewDraw. Ensure that you save the current project.

© 2008 QuickLogic Corporation

J.3 Creating a Schematic Design in ViewDraw

Using the tool's graphical interface, create the schematic design. For more information on how to create a design, refer to the ViewDraw online help.



Once you have finished your design in ViewDraw, check the design for errors and save it. Now your design is ready to be exported to EDIF netlist format.

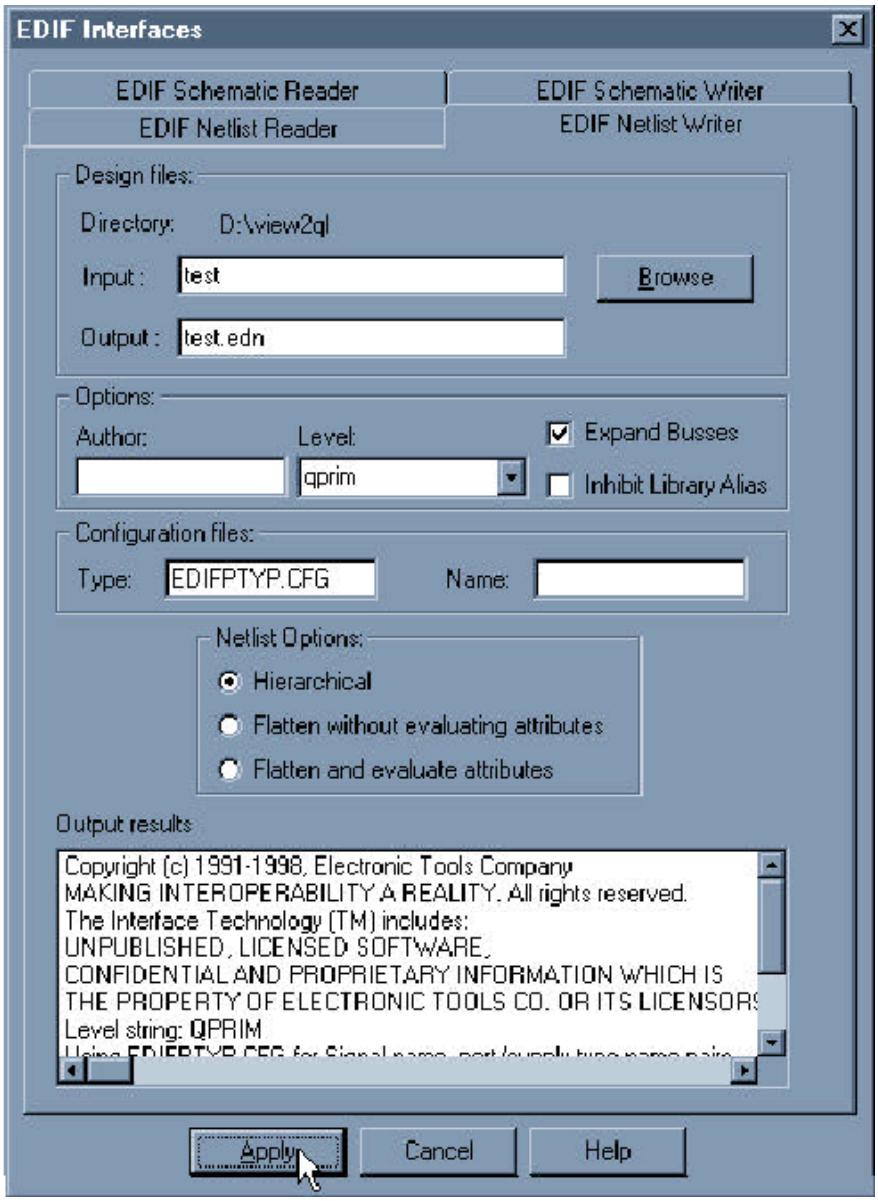
© 2008 QuickLogic Corporation

J.4 Exporting in EDIF

To export the EDIF netlist format:

1. Select **Tools>Export EDIF** from ViewDraw. The EDIF Interfaces dialog box opens.
2. On the EDIF Netlist Writer tab, select the **Expand Busses** box and to type `qprim` in the Level box.
3. Click **Apply**.

NOTE: For PCI32 Designs, use QMACRO instead of QPRIM in the Level box.



© 2008 QuickLogic Corporation

J.5 Import EDIF Netlist into SpDE

Before you import the EDIF lie into SpDE, select a default part and package:

1. Open `Spde.ini`, located at `..pasic\spde\data\`.
2. Change the values of the device and package to your desired part/package combination. You only need to do this once. The default values are as follows:

   ```
   DefaultEDIFPart=ql3025
   DefaultEDIFPackage=pq208
   ```

3. Save the `SpDE.ini` file.
4. Open SpDE and select **File>Open**.
   Now you can select the file `<your_design>.edn` and then run the tools.

NOTE: There are many other options available and techniques to explore for the designer who uses ViewLogic' s ViewDraw to QuickLogic design flow. However, this appendix is designed to give only a brief introduction to the design flow. For a more in-depth look at the available options, consult your ViewLogic ViewDraw and other QuickLogic documentation.

© 2008 QuickLogic Corporation

Appendix K
QCF File Constraints

This appendix provides an overview of constraint options that can be entered in the QuickLogic constraint (`.qcf`) file to enhance SpDE's performance as required. It contains the following sections:

- **"Introduction" on page 585**
- **"Constraint Options" on page 585**
- **"Conclusion" on page 605**

K.1 Introduction

Running the Place-and-Route tool in SpDE without constraints often provides optimal design performance. However, for certain designs, attaining the desired performance requires the ability to fix-place flip-flops, limit path-delays, or assign signals to particular pins. SpDE (version 9.2 and higher) provides this flexibility in the form of an editable QuickLogic constraint file and a Constraint Editor, accessible through SpDE by selecting **Tools>Constraints**.

NOTE: It is recommended that you use the Design Constraint Editor integrated in SpDE, as it eliminates the need to manually edit SpDE constraint files at various steps in the design flow.

This appendix explains the available constraint options and the constraint format to be followed when manually editing the `.qcf` file. The constraints listed here are supported by SpDE versions 9.2 and higher. To reduce delays caused by high-fanout gates, a description of buffering and duplication techniques is also provided.

K.2 Constraint Options

The following sections describe how to configure I/O banks and list the various constraint options and how each of the following constraints functions, including examples of each constraint format:

- **I/O Banks Configuration Constraints:**
  - bankstandard
  - padstandard
  - pullgate
  - pullset
  - bankslew
  - bankdrivestrengthP
  - setstandard
  - setslew
  - setdrivestrengthP

© 2008 QuickLogic Corporation

- setdrivestrengthN
- pullup
- bank: set bank voltage
- set: set set voltage

- **I/O Pad Related Constraints:**
  - place
  - pulldown
  - slew
  - unused I/O
  - fixhold
- **Timing Related Constraints:**
  - define_clock
  - path
  - set_arrival_time
  - set_departure_time
  - set_clock_to_out_delay
  - set_setup_time
  - set_false_path
  - set_multicycle_path
  - set_point_to_point_delay
- **Buffering/Fanout-Related Constraints:**
  - dont buffer
  - dup_on
  - dup_off
  - fanoutlimit
  - duplicate_all
  - high_fanout_net
  - high_fanout_net_limit
- **Flip-Flops, RAM Blocks, ECU Blocks, and Logic Modules Fix Placement Constraints:**
  - ffplacement
  - pull_all_ff_into_io
  - pull_ff_into_io
  - dont_pull_ff_into_io
  - placeecu
  - placeram
  - window_in_placer
  - assign_clk_to_quad
  - assign_enable_load

© 2008 QuickLogic Corporation

K.2.1 I/O Banks Configuration Constraints

Configuring I/O signaling standards in SpDE consists of two steps:

1. Designating the I/O banks to a specific standard.
2. Configuring the I/O ports to a specific standard.

During the Place-and-Route process, SpDE groups similar I/Os together and places them in compatible I/O banks.

K.2.1.1 Designating the I/O Bank Standard

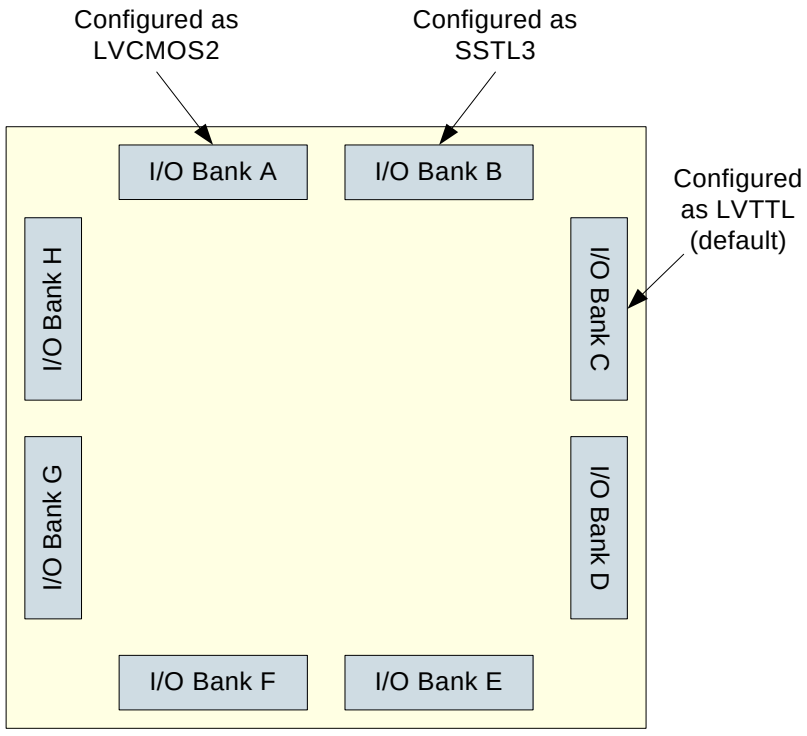
Format: `bankstandard {bank name} {bank standard}`

Examples:

```
bankstandard BANK_A LVCMOS2
bankstandard BANK_B SSTL3
bankstandard BANK_C LVTTL
```

Description: This constraint is used to assign a standard to an I/O Bank in the Eclipse and PolarPro II devices (see **Figure K-1**). The following I/O standards are available in Eclipse devices: LVTTL (default), LVCMOS2, GTL+, PCI, SSTL3, and SSTL2. The following I/O standards are available in the PolarPro II device: CMOS (default), LVCMOS, LVTTL and PCI.

Figure K-1: I/O Banks in Eclipse Devices



NOTE: When using a bank in GTL+, SSTL3, or SSTL2, the bank INREF pin must be tied to the appropriate value.

K.2.1.2 Configuring the I/O Port Interface Standard

Format: `padstandard {I/O port name} {LVTTL | LVCMOS2 | PCI | SSTL3 | SSTL2 | GTL+}`

Examples:

```
padstandard ecu_in[0] LVCMOS2
padstandard RAM_Dout[7] SSTL3
```

Description: This constraint is used to set the I/O standard for an I/O pin to interface with external devices (e.g., LVTTL, LVCMOS2, PCI, SSTL3, SSTL2, or GTL+).

K.2.1.3 Designating the Set Standard

Format: `setstandard {setname} {standardname}`

Examples:

```
setstandard SET1SSTL3
```

Description: This constraint is used to assign a standard to a set in the PolarPro device. Sets are sets of DDRIOs grouped together. For instance, Bank D (see **Figure K-2**) is divided into sets of 12 DDRs (see **Figure K-3**).

Figure K-2: VCCIO Bank Configuration

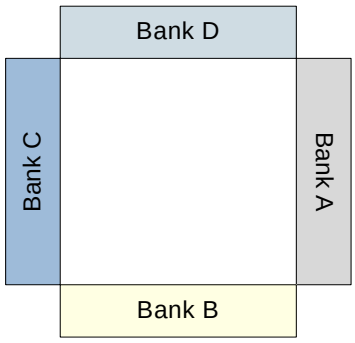
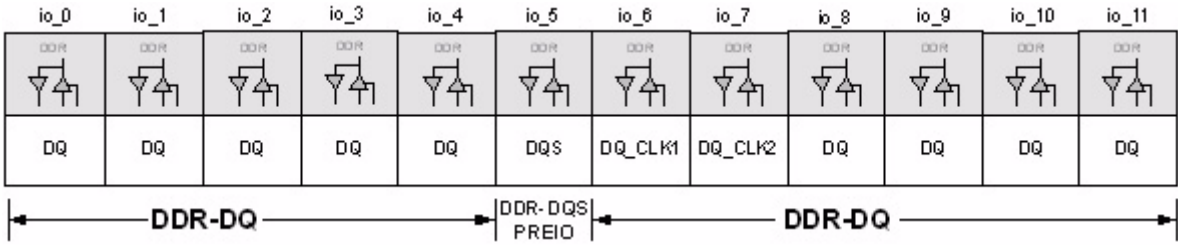


Figure K-3: DDR-IO set



NOTE: Before assigning the set standard, set should be assigned with gates using pullset constraints.

© 2008 QuickLogic Corporation

K.2.1.4 Assigning Gates to Banks

Format: `pullgate {bankname} {netname}`

Examples:

```
pullgate BANK_A N_1
pullgate BANK_B N_3
pullgate BANK_BN_5
```

Description: This constraint is used to assign gates to banks in PolarPro and PolarPro II devices.

K.2.1.5 Assigning Gates to Sets

Format: `pullset {setname} {netname}`

Examples:

```
pullset SET1 N_10
pullset SET2 N_11
```

Description: This constraint is used to assign gates to sets in the PolarPro device.

K.2.1.6 Designating Slew to Banks

Format: `bankslew {bankname} {slewname}`

Examples:

```
bankslew BANK_A slow
bankslew BANK_B fast
```

Description: This constraint is used to assign slew values to banks for the PolarPro device.

NOTE: Before assigning a slew value, banks should be assigned with gates using pullgate constraints.

K.2.1.7 Designating DriveStrengthP to Banks

Format: `bankdrivestrengthP {bankname} {drivestrengthvalue}`

Examples:

```
bankdrivestrengthP BANK_A 2
bankdrivestrengthP BANK_B 4
```

Description: This constraint is used to assign drive strength value to banks for the PolarPro device.

NOTE: Before assigning a drive strength value, banks should be assigned with gates using pullgate constraints.

K.2.1.8 Designating Slew to Sets

Format: `setslew {-type} {setname} {slewname}`

© 2008 QuickLogic Corporation

Examples:

```
setslew -dq set1 fast
setslew -dqs set2 slow
setslew -clock set2 fast
setslew set3 fast
```

Description: This constraint is used to assign slew values to sets for the PolarPro device. If the type is not specified, then GPIO is mapped as DDRIO. The slew value is set for that gate.

NOTE: Before assigning a slew value, sets should be assigned with gates using pullset constraints.

K.2.1.9 Designating DriveStrengthP to Sets

Format: `setdrivestrengthP {-type} {setname} {drivestrenghtvalue}`

Examples:

```
setdrivestrengthP -dq set1 10
setdrivestrengthP -dqs set2 2
```

Description: This constraint is used to assign drive strength values to sets for the PolarPro device. If the type is not specified, then GPIO is mapped as DDRIO. The drive strength value is set for that gate.

NOTE: Before assigning a drive strength value, sets should be assigned with gates using pullset constraints.

K.2.1.10 Designating DriveStrengthN to Sets

Format: `setdrivestrengthN {-type} {setname} {drivestrenghtvalue}`

Examples:

```
setdrivestrengthN -dq set1 10
setdrivestrengthN -dqs set2 2
```

Description: This constraint is used to assign drive strength values to sets for the PolarPro device. If the type is not specified, then GPIO is mapped as DDRIO. The drive strength value is set for that gate.

NOTE: Before assigning a drive strength value, sets should be assigned with gates using pullset constraints.

K.2.1.11 Setting Pullup

Format: `pullup netname} {TRUE|FALSE}`

Examples:

```
pullup N_1 TRUE
pullup N_10 FALSE
```

Description: This constraint is used to set a pullup value for the PolarPro device I/Os.

© 2008 QuickLogic Corporation

K.2.1.12 Setting Bank Voltage

Format: `bank {bankname} vccio {voltagevalue}`

Examples:

```
bank BANK_A vccio 2.5
bank BANK_B vccio 1.8
```

Description: This constraint is used to set a bank voltage value.

NOTE: Before assigning a bank voltage value, banks should be assigned with gates using pullgate constraints.

K.2.1.13 Setting Set Voltage

Format: `set {setname} vccio {voltagevalue}`

Examples:

```
set BANK_A vccio 2.5
set BANK_B vccio 1.8
```

Description: This constraint is used to set set voltage value.

NOTE: Before assigning a set voltage value, sets should be assigned with gates using pullset constraints.

K.2.2 I/O Pad Related Constraints

K.2.2.1 Fix-Placing I/O Signals to Pins

Format: `place {I/O port name} {I/O pin number}`

Examples:

```
# for BGA packages, pin numbering is a
# (row: alphabet)(column: number) scheme.
# using a BGA package
place qout[3] A17
place clk C10
# for other package types, the pin numbering begins with
# the string "IO" followed by the pin number
place qout[3] IO16
place clk IO22
```

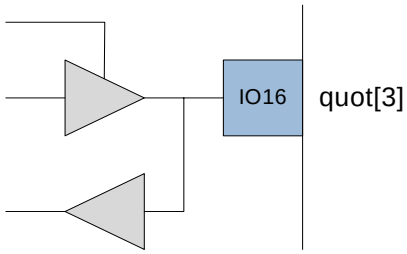
Description: This constraint is used for fix-placing I/O signals to pins (see **Figure K-4**).

NOTE: BGA package pins are labeled using the following format:
`[(row) alphabet: A, B, C…AA, BB,…][(column) number: 1,2,3…]`.

© 2008 QuickLogic Corporation

NOTE: QFP package pins are labeled using the following format: `I/O#` (e.g., `IO16`).

Figure K-4: Fixing Signals to Pins



K.2.2.2 Enabling the I/O Pull-Down Resistor

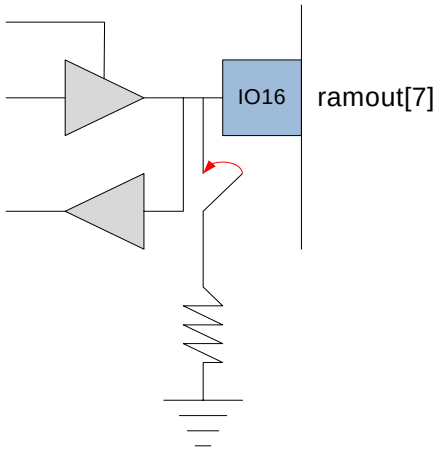
Format: `pulldown {net-name} {TRUE | FALSE}`

Examples:

```
pulldown ramout[7] TRUE
```

Description: In Eclipse and PolarPro II devices, every I/O pin has a programmable weak pull-down option (see **Figure K-5**). This constraint programs weak pull-downs to a specified I/O pad (default is FALSE). It is useful in situations when the I/O pin is not actively driven by an external device, but still needs to be held in a steady known state (in this case, LOW). This action prevents excessive current resulting from noise.

Figure K-5: Programmable Weak Pull-Down Resistor



K.2.2.3 Setting the Slew Rate for I/O Pads (Eclipse and PolarPro II Devices)

Format: `slew {net-name} {FAST | SLOW}`

Examples:

```
slew ecu_in[0] FAST
```

Description: This constraint sets the slew rate of an output signal to FAST or SLOW. The default slew rate is FAST.

© 2008 QuickLogic Corporation

K.2.2.4 Tying Unused I/Os to Default Values

Format: `UnusedI/O {VCC | GND | HIZ}`

Examples:

```
UnusedIO VCC
```

Description: This constraint is used to tie unused I/O pads internally to Vcc, Gnd or HiZ. The purpose of using this constraint is to minimize power consumption by fixing unused pads to Vcc or Gnd so that these pins do not float and draw excess current.

K.2.2.5 Setting the Fixhold for I/O Pads (PolarPro II)

Format: `fixhold {net-name} {TRUE | FALSE}`

Examples:

```
fixhold A5 FALSE
```

Description: This constraint sets the fixhold value for I/Os of the PolarPro II device.

K.2.3 Timing Related Constraints

K.2.3.1 Frequency, Setup, and Clock-to-Out Constraints

Format: `define_clock {clkName} –period {period} –setup {Tsu time} –clktoout {Tco time}`

Examples:

```
define_clock clk -period 25.0 -setup 5.0 -clktoout 4.0
```

Description: For every clock signal in the design, frequency, setup, and clock-to-out constraints can be given. The first argument is the desired clock period in nanoseconds (ns). For a clock frequency of 40 MHz, type 25.0. The second argument is setup time (e.g., 5.0 in the above example). Entering this constraint will assist the placer tool in finding the optimal placement of the input flip-flops. The third and last argument is the clock-to-out delay (e.g., 4.0 in the above example). The placer will try to place the output flip-flops associated with the specified clock as close as possible to the output pads so that the given Tco is satisfied. The constraints can be set on any nets defined in the design, but these constraints are honored by the tools only if the net is clock net or derived clock net.

K.2.3.2 Setting Path Timing Constraints

Format: `path {timing constraint} {start net-name} {through net-name1} {through net-name2}… {stop net-name}`

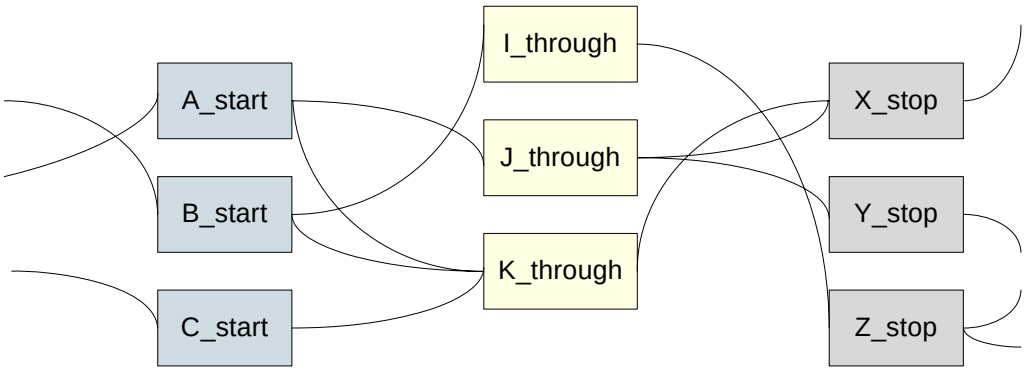
Examples:

```
path 22.0 B_start K_through X_stop
```

© 2008 QuickLogic Corporation

Description: This option is useful when there is a timing-critical path in the design. For example, in **Figure K-6**, if the delay for the path [B_start - K_through - X_stop] needs to be less than 22 ns, then the constraint in the `.qcf` file should look like the example above. The placer acknowledges the constraint and increases the priority for those nets on the specified path.

Figure K-6: Path Timing Constraints



K.2.3.3 Setting Arrival Times for Incoming Signals

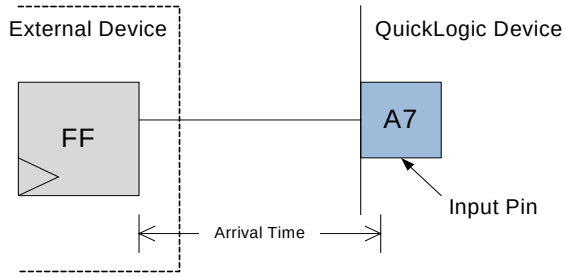
Format: `set_arrival_time {value} -clock {clkName} {net-name}`

Examples:

```
set_arrival_time 15.0 -clock clk start_z
```

Description: This constraint sets the arrival time (in nanoseconds) of a signal from the output of an external device to the input pin of the QuickLogic device with respect to the specified clock domain (see **Figure K-7**). The `net-name` is the input pin signal name.

Figure K-7: Arrival Time Definition



K.2.3.4 Setting Departure Times for Outgoing Signals

Format: `set_departure_time {value} -clock {clkName} {net-name}`

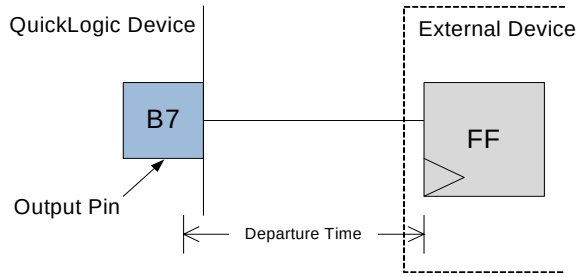
Examples:

```
set_departure_time 15.0 -clock clk qout_z
```


© 2008 QuickLogic Corporation

Description: This constraint sets the departure time of a signal (in nanoseconds) from the output pin of the QuickLogic device to the input pin of an external device with respect to the specified clock domain (see **Figure K-8**). The `net-name` is not the output pin signal name; it is the name of the input net to the output buffer of the I/O pin (usually given by SpDE).

Figure K-8: Departure Time Definition



K.2.3.5 Setting Clock-to-Out Delays from a Flip-Flop to an Output Pin

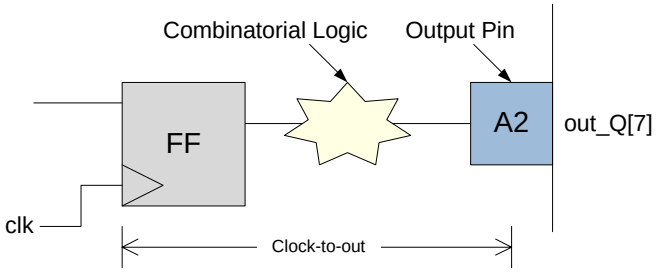
Format: `set_clk_to_out_delay {value} -clock {clkName} {net-name}`

Examples:

```
set_clk_to_out_delay 10.0 -clock clk out_Q[7]
```

Description: Sets a clock-to-out constraint (in nanoseconds) for the specified output signal with respect to the specified clock domain (see **Figure K-9**).

Figure K-9: Clock-to-Out Delay



K.2.3.6 Placing SetupTime Constraints for an Incoming Signal to a Flip Flop

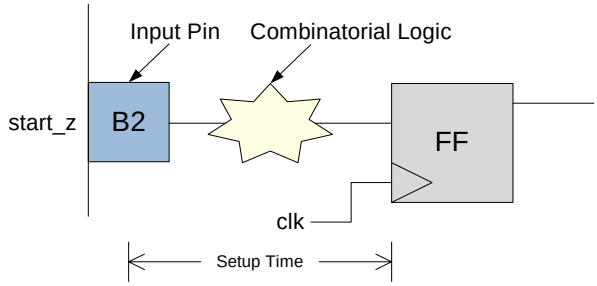
Format: `set_setup_time {value} -clock {clkName} {net-name}`

Examples:

```
set_setup_time 5.0 -clock clk start_z
```

Description: Places a setup time constraint (in nanoseconds) for the specified input signal with respect to the specified clock domain (see **Figure K-10**).

Figure K-10: Setup Time Definition



K.2.3.7 Setting False Paths—Nets Not Included in Frequency Calculation

Format:

```
set_false_path –from {net-name}
set_false_path –from {net-name} –through {net-name}
set_false_path –from {net-name} –to {net-name}
set_false_path –from {net-name} –to {net-name} –through {net-name1, net-name2, netname3,…..}
set_false_path –to {net-name}
set_false_path –to {net-name} –through {net-name}
```

Examples:

```
set_false_path -from abc -to xyz
set_false_path -from abc -to xyz -through ijk
```

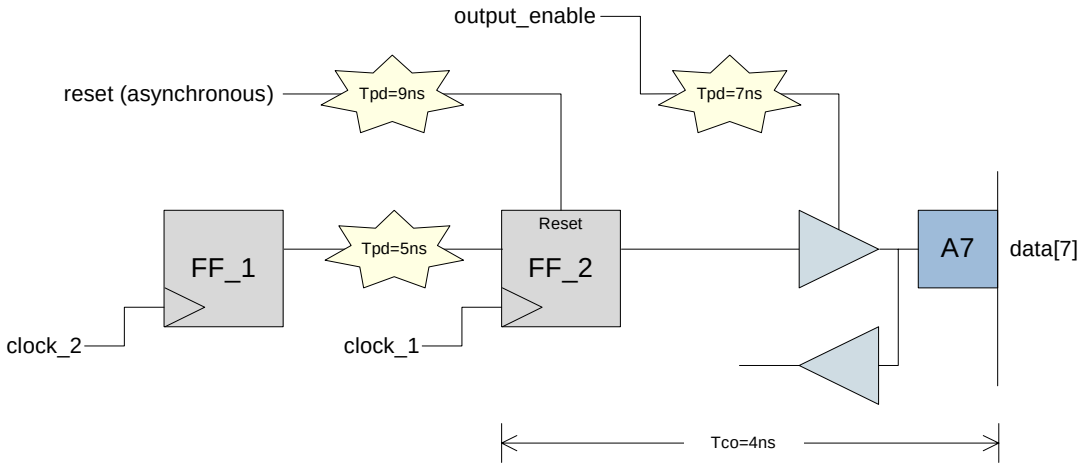
Description: False paths are non-critical paths in the design that can be ignored when performing timing analysis on the design. Some examples of false paths are slow output enable signals, synchronous reset signals, and paths between two flip-flops driven by clocks that are synchronous with respect to one another. In **Figure K-11**, the following three examples are illustrated:

- If clock_1 and clock_2 are asynchronous with respect to each other, then the path from the output of FF_1 to the input of FF_2 (with Tpd = 5 ns) can be considered a false path.
- Assume that clock_1 and clock_2 are synchronous (or have the same driver). The propagation delay from the output of FF_1 to the input of FF_2 is 5 ns. The asynchronous reset signal has a propagation delay of 9 ns. Timing analysis would show that the longest path in the design is through the reset signal, and hence Fmax would be 1/9 ns. In this case, the path through the reset signal can be considered a false path.
- Consider the situation where clock_1 and clock_2 are synchronous (or have the same driver) and the reset signal for FF_2 is synchronous. If the output_enable signal is tied HIGH (enabled), then Tco = 4 ns. But if output_enable is driven by an internally generated signal

© 2008 QuickLogic Corporation

that has a propagation delay of 7 ns, then the timing analyzer would report that Tco = 7 ns, which is incorrect. Along these lines, the path through the output_enable signal can be considered a false path.

Figure K-11: Examples of False Paths in Designs



K.2.3.8 Setting Multi-Cycle Paths—Nets with Multi-Cycle Propagation Delay

Format:

```
set_multicycle_path {# of clock cycles} –from {net-name}
set_multicycle_path {# of clock cycles} –from {net-name} –to {net-name} -through {netname1, net-name2, net-name3…}
set_multicycle_path {# of clock cycles} –from {net-name} –through{net-name}
set_multicycle_path {# of clock cycles} –from {net-name} –to{net-name}
set_multicycle_path {# of clock cycles} –to {net-name}
set_multicycle_path {# of clock cycles} –to {net-name} –through{net-name}
```

Examples:

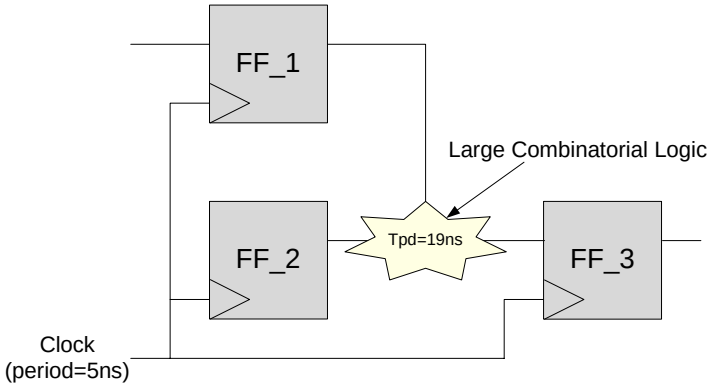
```
Set_multicycle_path 3 -from Az -to Bz
```

Description: A multi-cycle path is a path that requires more than one clock cycle to propagate. In **Figure K-12**, the clock supplied to the three flip-flops has a period of 5 ns. The outputs of FF_1 and FF_2 enter a large combinatorial logic block (specified by design) that has a maximum path delay of 19 ns. The output of this combinatorial block is then registered by FF_3. In this case, these two paths (nets from output of FF_1 to FF_3 and FF_2 to FF_3) are multi-cycle paths.

© 2008 QuickLogic Corporation

In this particular example, the paths previously mentioned require 3.8 (19 ns/5 ns) clock cycles. Rounded to the next highest integer, this gives four clock cycles. The timing analyzer will treat this path as having 3.8 ns of propagation time.

Figure K-12: Multi-cycle Paths



K.2.3.9 Setting Point-to-Point Timing Constraints

Format: `set_point_to_point_delay {delay} –from {start flip-flop output} –to {end flip-flop output}`

Examples:

```
set_point_to_point_delay 5.0 -from Az -to Bz
```

Description: Places a delay constraint (in nanoseconds) between any two points. Start and stop points include: flip-flops, I/O cells, RAM, and ECU blocks. This constraint is useful for reducing path delays on critical nets.

K.2.4 Buffering Techniques

Buffering and duplication methods are essentially solutions for the same problem – high-fanout nets. Gates that have high fanout reduce the overall performance of a design because of the delay introduced in driving large capacitive loads. Buffering nets driven by high-fanout gates reduces capacitive loading on those nets. Buffering can be implemented using the following two methods:

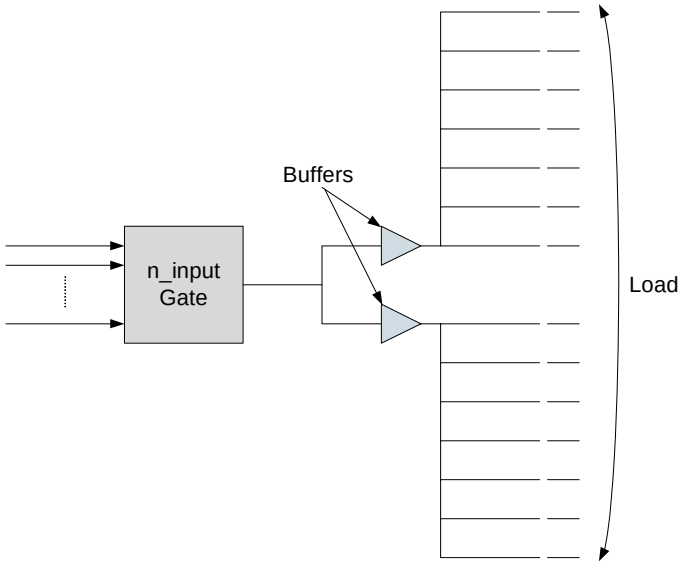
- Split buffering
- Selective buffering

© 2008 QuickLogic Corporation

K.2.4.1 Split Buffering

Split buffering divides the net of the high-fanout gate into two or more nets and then inserts buffers on those newly created nets (see **Figure K-13**).

Figure K-13: Split Buffering

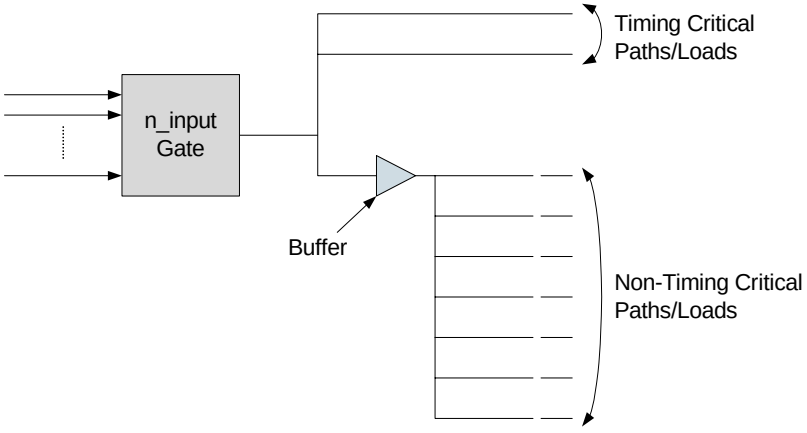


Initially, without split buffering, the n-input gate drives 14 loads. After split buffering, the n-input gate drives two loads and each inserted buffer drives a fanout of seven loads.

K.2.4.2 Selective Buffering

Selective buffering is most effective when only a few paths/loads on a gate's output are timing-critical while the rest are not critical (see **Figure K-14**).

Figure K-14: Selective Buffering



Two of the output nets are in the critical path, the rest are not timing-critical. It is important to note that inserting buffers boosts signal strength and reduces the capacitive loading on the driving gate, but at a cost: the buffer propagation delay. In this case, inserting buffers on two

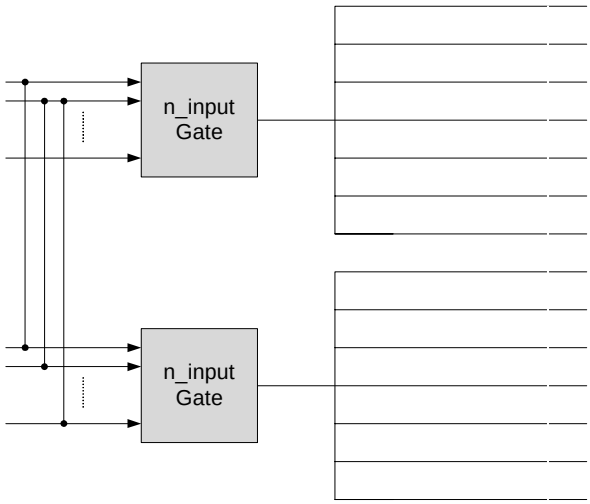
© 2008 QuickLogic Corporation

of the timing-critical paths increases the path propagation delay. Buffering only the non-critical paths reduces the load driven by the n-input gate and the time taken by the signal to propagate to the two timing-critical loads.

K.2.4.3 Duplicating Gates

An alternative to buffering gates is the duplication of the driver gates (see **Figure K-15**). Using this method avoids the propagation delay introduced by buffers. The drawbacks are that the input nets have increased fanout and the duplicated gates take up additional resources and possibly clog up routing and logic cell resources.

Figure K-15: Duplicating Gates



As shown in **Figure K-15**, before duplication, an n-input gate has a fanout of 14. After duplicating the n-input gate, each of the two resulting n-input logic blocks drives seven loads.

K.2.5 Buffering/Fanout-Related Constraints

K.2.5.1 Prohibiting Buffering on Nets

NOTE: The following six constraints are interrelated and deal with duplicating gates. Read the contents in all six tables before starting to concentrate on one constraint.

Format: `dont buffer {net-name1} {net-name2}`

Examples:

```
dont buffer clockz clearz
```

Description: This constraint prohibits SpDE's auto-buffering algorithm from placing buffers on the specified nets. The net-names specified must be internal net-names and not pin names. This constraint is useful in situations where a net has high-fanout but is not performance critical. Consequently, placing the `dont buffer` constraint on such nets frees up logic cell resources for buffering other timing critical nets.

© 2008 QuickLogic Corporation

K.2.5.2 Duplicating Drivers with High Fanout

Format: `dup_on`

Examples:

```
dup_on
duplicate_all
fanoutlimit 10
dup_on
fanoutlimit 10
high_fanout_net I1.N_48
```

Description: This constraint allows SpDE to duplicate drivers with high fanout. It should be used to turn the duplication feature ON. This works for all logic, except flip-flops. The following constraints are turned ON with the `dup_on` constraint:

```
- duplicate_all
- high_fanout_net
- high_fanout_net_limit
```

K.2.5.3 Turning Off the Duplication Feature

Format: `dup_off`

Examples:

```
dup_off
fanoutlimit 10
high_fanout_net I1.N_48
```

Description: This constraint is used to turn OFF the net duplication feature. Instead of deleting the net duplication constraint, if `dup_off` is included in the `.qcf` file, all duplication constraints will be neglected. This constraint eliminates the process of deleting and retyping (or commenting out) duplication constraints in the `.qcf` file. The following constraints are turned OFF with the `dup_off` constraint:

```
- duplicate all
- high_fanout_net
- high_fanout_net_limit
```

K.2.5.4 Setting Fanout Limits on All Nets

Format: `fanoutlimit {fanout limit number}`

Examples:

```
fanoutlimit 9
duplicate_all
high_fanout_net ffout_z
```

© 2008 QuickLogic Corporation

Description: This constraint sets the global fanout limit in the design for specified nets using the following duplication constraints:

```
- duplicate all
- high_fanout_net
```

K.2.5.5 Duplicating Nets

Format: `duplicate_all`

Examples:

```
duplicate_all
```

Description: This is a global constraint to duplicate all gates with fanouts greater than the fanout limit specified by the above `fanoutlimit` constraint. If `fanoutlimit` is not specified, then the default limit is 16.

K.2.5.6 Duplicating Specific Nets Using a Global Fanout Limit

Format: `high_fanout_net {netName}`

Examples:

```
high_fanout_net I1.N_48
```

Description: This constraint duplicates the driver gate of the specified net if it has a fanout greater than the specified fanout limit.

K.2.5.7 Duplicating Specific Nets Using a Specific Fanout Limit

Format: `high_fanout_net_limit {netName} {fanout limit}`

Examples:

```
high_fanout_net_limit I1.N_48 7
```

Description: This constraint duplicates the driver gate of the specified net if that same driver gate has a fanout greater than the specified fanout limit.

K.2.6 Flip-Flops, RAM Blocks, ECU Blocks, and Logic Modules Fix Placement Constraints

K.2.6.1 Fixing Flip-Flop Placement

Format: `ffplacement {flip-flop output net-name} {Logic Cell}`

Examples:

```
ffplacement qoutz[0] A12
ffplacement qoutz[1] A16
```

Description: This constraint fixes a flip-flop to a logic cell. Find the net-name of the flip-flop output signal from SpDE's chip layout and choose the logic cell that it should be fixed to. The logic cell name is found below the logic cell block in the chip layout window in SpDE.

© 2008 QuickLogic Corporation

NOTE: (For Eclipse devices): This constraint only works for the Q1 flip flop; the feature to fix place the Q2 flip flop is currently not available. In addition, this constraint to fix flip flops can only be used for internal logic cells, not I/O flip-flops.

K.2.6.2 Pulling ALL Flip-Flops Into I/O Cells

Format: `pull_all_ff_into_io`

Examples:

```
pull_all_ff_into_io
```

Description: Allows SpDE to pull flip-flops registering data (input or output) into I/O cells (if possible). This constraint allows setup time and clock-to-out delays to be minimized.

K.2.6.3 Pulling a Flip-Flop Into an I/O Cell

Format: `pull_ff_into_io {flip-flop output net-name}`

Examples:

```
pull_ff_into_io out_Qz[8]
```

Description: Drags the specified flip-flop into the I/O pad cell. This feature supports input, output and output enable flip-flops. When targeting devices that do not have output flip-flops in the I/O pad cells, SpDE will place the output flip-flop as close to the I/O as possible.

K.2.6.4 Prohibiting the Use of I/O Flip-Flops

Format: `dont_pull_ff_into_io {flip-flop output net-name}`

Examples:

```
pull_all_ff_into_io
dont_pull_ff_into_io dataout_z[5]
```

Description: This constraint is normally used in conjunction with `pull_all_ff_into_io`.

K.2.6.5 Fix-Placing ECU Blocks

Format: `placeecu {any data output signal belonging to the ECU block} {ECU cell name}`

Examples:

```
placeecu q_ecuz[0] ECU_H
```

Description: This constraint is used for fix-placing ECU blocks to ECU cells on the chip.

K.2.6.6 Fix-Placing RAM Blocks

Format: `placeram {any data output signal belonging to the RAM block} {NETNAME} {RAMCELLNAME} {fixType}`

`NETNAME`—Net name of the RAM

`RAMCELLNAME`—RAM cell name

`fixType`—Applicable to the PolarPro device only. The values should be ODD or EVEN.

© 2008 QuickLogic Corporation

Examples:

```
placeram rdz[0] RAM_E1
placeram rdz[0] RAM2_A1 ODD (for polarpro devices)
```

Description: This constraint is used for fix-placing RAM blocks to RAM cells on the chip.

K.2.6.7 Fix-Placing Logic Modules

Format: `window_in_placer {module name} {upper- left logic cell} {lower-right logic cell}`

Examples:

```
window_in_placer I225 A1 AB2
window_in_placer ALLGATES A1 AB2
```

Description: For more information on this feature refer to Application Note 61 at http://www.quicklogic.com/images/appnote61.pdf.

K.2.6.8 Manual Clock Assignment

Purpose: To place the given clock net loads into the specified quad or quads.

Format: `assign_clk_to_quad {clock net name} {comma separated quad names}`

`{Clock net name}—The output net of the clock pad.`

`{Comma separated quad names}—The designer can specify one or more quads separated by comma. All devices except the PolarPro QL1P1000 have four quads. The PolarPro QL1P1000 device has 16 quads.`

QL1P1000 device:

QUAD01	QUAD02	QUAD05	QUAD06
QUAD03	QUAD04	QUAD07	QUAD08
QUAD09	QUAD010	QUAD013	QUAD014
QUAD011	QUAD012	QUAD015	QUAD016

All other devices have only four quads.

QUAD01	QUAD02
QUAD03	QUAD04

Examples:

```
assign_clk_to_quad abc QUAD01, QUAD02,QUAD03
assign_clk_to_quad N_12 QUAD14
```

Description: This constraint is used to control the clock load placement at quadrant level. During the clock load placement the placer tool will try to do a clock load placement as a user-specified constraint.

© 2008 QuickLogic Corporation

K.2.6.9 Clock Enable Assignment

Purpose: Assign device column clocks to the design clock enable loads. This constraint is applicable for PolarPro II devices only.

Format: `assign_enable_load {enable name} load {load} {column name} {quad name}`

`{Enable name}`—The clock enable output net name.

`{load}`—This is the count of clock enable loads that goes into device specified clock columns. This should be an integer.

`{column name}`—The column names (A,B..Z,AB,AC..). Refer the below diagram for column names.

`{quad name}`—The quad names (QUAD01, QUAD02, QUAD03 and QUAD04)

For PolarPro II device QL2P150:

QUAD01 A..R	QUAD02 S..Z AA..AJ
QUAD03 A..R	QUAD04 S..Z AA..AJ

NOTE: The column names are formed by removing the number portion of logic cell name (e.g., if the logic cell name is AB1, then the column name is AB).

Examples:

```
assign_enable_load N_18 load 10 A QUAD03
assign_enable_load abc load 22 AC QUAD02
assign_enable_load N_12 load 5 AG QUAD04
```

K.2.6.10 CCM Placement

Format: `placeccm {NETNAME} {CCMNAME}`

`NETNAME`—Net connected to pllout0 port of CCM

`CCMNAME`—CCM Cell Name (i.e., CCM_TR or CCM_TL)

Examples:

```
placeccm N_6 CCM_TR
```

Description: This constraint is used to place CCM, and is applicable to the PolarPro device only.

K.3 Conclusion

This main purpose of this appendix is to provide a comprehensive list of the available design constraint options and constraint formats that should be used in the `.qcf` file to assist in design performance optimization. These constraints can be entered by using the graphical interface

© 2008 QuickLogic Corporation

in SpDE or by manually editing the QuickLogic constraint (`.qcf`) file. Fanout-related constraints do not have a graphical interface in SpDE, so the `.qcf` file should be edited to incorporate this feature.

CAUTION: The `.qcf` file cannot be edited while SpDE is reading the file. Keep a backup copy of modified `.qcf` files.

© 2008 QuickLogic Corporation

Index

A

Active HDL
- functional simulation **24**
- in design flow **23**
- schematic-based design flow **181**
- Simulator, using **18**, **20**
- user interface **197**
- Verilog design flow **186**
- VHDL design flow **187**

Adders, overview **447**

Attributes
- adding in SCS Hierarchy Navigator **151**
- adding in SCS Schematic Editor **151**
- fixed **182**
- pack **140**, **182**
- place **182**
- synmacro **182**

Autobuffering
- how it works **121**
- in SpDE **122**

Automatic buffering tools **121**

B

Back Annotation **174**

Buffering. *See* Design Techniques

C

Cell, QDIF definition of **523**

Clock Networks
- global **159**
- in Eclipse Devices **53**

Combinatorial Macros, overview **455**

Comparitors, overview **447**

Constraint Manager
- overview **267**
- using in SpDE **268**
- Window-based placement editor **280**

Corner. *See* Delay Modeler

Counters
- down counters **441**
- up counters **435**
- up/down counters **444**

Custom Operating Range. *See* Delay Modeler

D

Delay Modeler
- corner **173**
- operating range **172**
- out-pad load **173**
- overview **172**
- setting custom operating range **174**

Design Constraint Editor **287**

Design Flows
- overview **179**
- schematic **180**
- Verilog-only **185**
- VHDL-only **187**

Design Techniques
- disabling autobuffering **122**
- double-buffering **127**
- inserting buffers **125**
- inserting buffers in VHDL/Verilog **128**
- paralleling **126**
- pipelining **130**
- selective buffering **125**
- SpDE autobuffering **122**
- split buffering **125**
- using automatic buffering **121**

Design Verifier, overview **366**

Device Registers, 0.25um overview **427**

Double-Buffering
- recommended limit **162**

Double-Buffering, overview **127**

E

Eclipse Devices
- clock networks **53**
- dedicated clock network **55**
- ECU modules **61**
- global clock networks **56**
- high drive networks **58**
- I/O standards **59**
- using Configuration Editor **69**

Eclipse-Specific Macros, overview **505**

Error Messages, reference **529**

© 2008 QuickLogic Corporation

F

Fanout. *See* Design Techniques
FIFO
creating modules **228**
design files **231**
File constraints
QCF file **585**
File Extensions, reference **389**
Fixed Placement
bussed pads **183**
flip-flops in schematic designs **152**
of I/Os in schematic designs **151**, **153**
of I/Os in Verilog designs **152**, **186**
overview **151**
Flip Flops, in macros **408**

G

Gates
AND gates **404**
NAND gates **404**
NOR gates **404**
OR gates **404**
overview **404**
QDIF definition of **523**
GND net **190**
GUI, QuickWorks design flow **22**

H

Hard Macros. *See* Macros
HDPADs. *See* High Drive Pads
High Drive Pads
overview **399**
recommended limit **162**
routing with **162**
Highlight Net
collapsing window **310**
mode in SpDE **310**
Pan to Net Driver **310**

I

I/O Configuration
in PolarPro devices **272**
I/O Pads **396**
high drive pads (HDPADs) **399**
how to use **182**
overview **396**
unused **308**
Installation
hardware requirements **5**
installing QuickWorks **6**
software requirements **5**
Instances. *See* SCS Schematic Editor
Interconnect
clock networks **159**
quad wires **161**
wire coding **354**
Interface, QuickWorks design flow **22**
Invalid Instance Name. *See* SCS Hierarchy Navigator

L

Latches, overview **416**
Leonardo Spectrum
generating EDIF output **559**
overview **556**
settings **559**
TCL script support **558**
writing HDLs for **559**
Logic Optimizer
area and speed optimization **139**
automatic buffering **141**
optimization modes **139**
simulation difficulties **139**
using in SpDE **137**

M

Macro Design Mode, setup in SpDE **202**
Macro Library. *See* pASIC Macro Library
Macros
constraints **206**
creating designs, schematic **207**
creating designs, Verilog or HDL **208**
creating in QuickWorks **201**
error messages **222**
fix placement constraints **219**
hard macros **395**
importing designs in SpDE **209**
importing designs to SpDE **202**
Macro Planner tool **212**
pASIC3, QuickRAM, and QuickPCI-Specific **497**
pASIC3, QuickRAM, QuickPCI, Eclipse, PolarPro, and PolarPro II-Specific **495**
processing designs **203**, **221**
saving design as macro **204**
setting up SpDE **202**
Master Cells
overview **494**
Mixed-mode designs **181**
Multiplexers, overview **421**
Multipliers, overview **447**

© 2008 QuickLogic Corporation

N

Net Name Wildcards, setting 290
Net, QDIF definition of 523
Netlist, choosing type 184
Nets
  highlighting in SpDE 310
  routing 155
  tying to VCC or GND 190

O

Operating Range. *See* Delay Modeler
OrCAD 349
Out-Pad Load. *See* Delay Modeler

P

Pads. *See* I/O Pads.
Pan to Net Driver. *See* Highlight Net
pASIC Macro Library
  0.25um device registers 427
  arithmetic 447
  combinational 455
  down counters 441
  Eclipse-specific macros 505
  flip-flops 408
  I/O pads 396
  latches 416
  muxes 421
  overview 394
  PolarPro-specific macros 512, 517
  QuickPCI-specific macros 504
  registers 424
  shifters 432
  simple gates 404
  TTL 489
  up counters 435
  up/down counters 444
Path Analyzer 323
  using timing constraints 149
Path-Constraint-Driven Placer. *See* Placer
PCI Configuration, using in SpDE 256
Physical Viewer, wire color coding 354
Pins, description of 396
Pipelining. *See* Design Techniques
Placer
  fixing placement 151
  overnight mode 147
  overview mode 143
  placement seed 147
  preliminary mode 147
  quality mode 147
  timing driven placer 149
PLL/CCM Wizard
  configuring PLL/CCM 263
  output settings 266
  running in SpDE 262
  selecting device family 262
  using in QuickWorks 261
PolarPro Devices
  CCM (Configurable Clock Manager) 88, 97
  clock network 87, 97
  clock pad configuration 88, 97
  configuring I/Os using GUI 274
  features and benefits 83, 93
  I/O bank configuration 85, 272
  logic cell design 92, 102
  overview 83
  RAM/FIFO implementation 90, 99
PolarPro II Devices
  overview 93
PolarPro II. *See* PolarPro II Devices
PolarPro. *See* PolarPro Devices
PolarPro-Specific Macros, overview 512, 517
Port, QDIF definition of 523
Power Simulator
  environment setup 111
  using in QuickWorks 111, 112

Q

QCF file constraints
  constraint options 585
  overview 585
QDIF
  creating netlist 184
  description of format 523
QuickLogic Data Interchange Format. *See* QDIF
QuickWorks
  creating macros 201
  schematic flow 180
  toolkit features 179
  Verilog design flow 185
QuickWorks design flow
  enabling and disabling 23
  using 23
QuickWorks icons, reference 551

R

RAM Blocks
  creating 225
  in QuickRAM devices 461
RAM Design Files 228
Registers, overview 424

© 2008 QuickLogic Corporation

ROM, creating ROMs 232
Router
and high-drive nets 162
interconnect resources 158
logic jog issue 169
net antenna issue 168
seed 155
tool options 155
using manual Routing Editor 163
using Priority Router 156
Routing Editor
adding route on net 166
deleting route on net 165
error messages 167
Routing. *See* Interconnect

S

SCS
SCS Executive 189
Symbol Editor 191
tools 189
SCS Hierarchy Navigator
adding attributes 151, 191
choosing netlist type 184
design flow 191
exporting QDIF 192
exporting Verilog 192
fixing pins and flip-flops 151
invalid instance 191
saving part selection 192
tree (.tre) files 191
SCS Schematic Editor & Navigator
adding pack attributes 140
adding place attributes 151
adding VCC and GND 190
connecting busses 190
design flow 189
fixing pins and flip-flops 151
grounding busses 190
iterated instances 190
launching 189
using busses 190
Seamless pASIC Design Environment. *See* SpDE
Sequencer, overview 177
Shift Registers, overview 432
Simulation
selecting a simulator 174
using optimization 139
Soft Macros, overview 395
SpDE
Active HDL Interface 197
autobuffering in 122
Design Constraint editor 287
highlight net mode 310
overview 351
pan to net driver 310
tool options 378
view preferences 371
Static Timing Analyzer
overview 319
Path Analyzer 323
toolbar 323
using in SpDE 320
Subtractors, overview 447
Symbols, creating bus pins for Verilog or VHDL 195
Synario Capture System. *See* SCS

T

TCL commands, used by SpDE GUI 36
TCL Interface
command line options 25
in QuickWorks 31
TCL scripts, running from SpDE 36
Term, QDIF definition of 523
Timing Constraints Editor, using in SpDE 288
Timing Constraints, in placement 149
Tools
Back Annotation 174
Delay Modeler 172
Design Verifier 366
Leonardo Spectrum 556
Macro Planner 212
Placer 143
PLL/CCM Wizard 261
Power Simulator 111
Router 155
selecting tools to run 378
Sequencer, description of 177
Tree (.tre) files. *See* SCS Hierarchy Navigator
TTL Macros, overview 489
Turbo Writer
launching 195
syntax checking 196

V

VCC net 190
Verilog
back annotating 175
creating files 195
creating netlist 184
creating symbols for 194
inserting buffers in 128
primitive file 175
syntax checking 196

© 2008 QuickLogic Corporation

Verilog-only design flow **185**
VHDL
creating files **195**
creating netlist **184**
creating symbols for **194**
inserting buffers in **128**
syntax checking **196**
VHDL-only design flow **187**
Viewsim, back annotation **176**

W

Web Update
installing devices and updates **388**
Window-Based Placement Editor, overview **280**

© 2008 QuickLogic Corporation

© 2008 QuickLogic Corporation